UNIVERSIDADE REGIONAL INTEGRADA DO
ALTO URUGUAI E DAS MISSÕES
URI – CAMPUS DE ERECHIM

DEPARTAMENTO DE ENGENHARIAS E CIÊNCIA DA COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM INFORMÁTICA

MANIPULANDO BANCO DE DADOS VIA INTERNET COM FERRAMENTAS GRATUITAS

Lucas Lopes dos Santos
Ronaldo do Amaral

Prof. Jacques Duílio Brancher
ORIENTADOR

ERECHIM/RS, JULHO DE 2000

MANIPULANDO BANCO DE DADOS VIA INTERNET COM FERRAMENTAS GRATUITAS

Trabalho apresentado como requisito à obtenção do grau de Bacharel em Informática pela Universidade Regional Integrada do Alto Uruguai e das Missões – URI, Campus de Erechim sob orientação do Professor **Jacques Duílio Brancher**

ERECHIM/RS, JULHO DE 2000

SUMÁRIO

SUMÁRIO ..... **IV**

RESUMO ..... **VII**

ABSTRACT ..... **VIII**

LISTA DE ABREVIATURAS ..... **IX**

ÍNDICE DE FIGURAS ..... **X**

ÍNDICE DE TABELAS ..... **XI**

INTRODUÇÃO **12**

1. LINUX **14**

1.1. UM BREVE HISTÓRICO DO LINUX 14
1.2. DISTRIBUIÇÕES DO LINUX 15
1.3. INSTALAÇÃO DO LINUX 16

2. MYSQL **18**

2.1. O QUE É MYSQL? 18
2.2. BREVE HISTÓRICO 18
2.3. POR QUE MYSQL? 19
2.4. PRINCIPAIS CARACTERÍSTICAS 19
2.5. POLÍTICA DE USO NA WEB 20
U$ 40 20
U$ 25 20
2.6. INFORMAÇÕES BÁSICAS SOBRE INSTALAÇÃO. 21
2.7. COMPATIBILIDADE DO BANCO MYSQL COM A LINGUAGEM SQL 22
2.8. TRANSAÇÕES 23
2.9. STORE PROCEDURES E TRIGGERS 24
2.10. SEGURANÇA E PRIVILÉGIOS DE ACESSO 24
2.10.1. User names e password *24*
2.11. COMO FUNCIONA A CONEXÃO AO MYSQL 26
2.12. PRINCIPAIS FERRAMENTAS PARA UTILIZAÇÃO DO MYSQL 27
2.12.1. MySQL *27*
2.12.2. Mysqladmin *27*

3. PERSONAL HOME PAGE(PHP) **30**

3.1. O QUE É PHP ? 30
3.2. HISTÓRICO 31

3.3. BANCOS DE DADOS 32
3.4. INSTALAÇÃO EM SISTEMAS UNIX 33
3.5. CONFIGURAÇÃO 35
3.6. SINTAXE BÁSICA 36
3.6.1. Delimitadores *36*
3.6.2. Separador de Instruções *37*
3.6.3. Definição de Variáveis *37*
3.6.4. Comentários *37*
3.7. EXEMPLOS 38
3.7.1. Um exemplo simples *38*
3.7.2. Exemplo de PHP com formulários HTML *39*
3.7.3. Exemplo do PHP acessando uma base de dados *40*

4. APACHE WEB SERVER **42**

4.1. O QUE É UM SERVIDOR WEB 42
4.2. O SERVIDOR APACHE 42
4.3. BREVE HISTÓRICO 43
4.4. INSTALAÇÃO 43
4.4.1. Os módulos do Apache *44*
4.4.1.1. O módulo PHP 44
4.5. CONFIGURAÇÃO 45
4.5.1. Arquivos de configuração *45*
4.5.1.1. Httpd.conf 45
4.5.1.2. Srm.conf 45
4.5.1.3. Access.conf 46
4.6. PROTOCOLO SSL (SECURE SOCKET LAYER) 46

5. IMPLEMENTAÇÃO **48**

5.1. FLUXOGRAMA 49
5.1.1. Descrição detalhada dos módulos *50*
5.1.1.1. Módulo 1 – Fazer Prova Virtual 50
5.1.1.2. Módulo 2 – Busca Rápida 51
5.1.1.3. Módulo 3 – Consultar Questões 51
5.1.2. Descrição do Fluxograma *52*
5.2. MODELAGEM DOS DADOS 53
5.2.1. Descrição das Tabelas *53*
5.2.2. Modelo E-R (Entidade - Relacionamento) *54*
5.2.3. Dicionário de Dados *55*
5.3. SINTAXE SQL PARA CRIAÇÃO DAS TABELAS 55
5.4. DESCRIÇÃO DAS FUNÇÕES PHP UTILIZADAS 56
5.5. IMPLEMENTAÇÃO PHP E HTML 58

5.5.1. Index.html – Home Page ... *58*
5.5.2. Prova.html ... *59*
5.5.3. Consulta.html ... *60*
5.5.4. Prova.php3 ... *61*
5.5.4.1. Concatenação das variáveis ... 64
5.5.5. Consulta.php3 ... *65*
5.5.6. Corrige.php3 ... *68*
5.5.7. Buscarap.php3 ... *69*

CONCLUSÕES ... **71**

REFERÊNCIAS BIBLIOGRÁFICAS ... **73**

Resumo

O presente trabalho tem como objetivo primordial definir um conjunto de ferramentas gratuitas para plataforma Linux que possam cobrir todas as etapas do desenvolvimento de um Web Site com manipulação de uma base de dados. Partiu-se inicialmente para o estudo de quais seriam as ferramentas necessárias e disponíveis para a instalação de um servidor Web. A partir disto foi implementado um Site para simulação de prova de vestibular. Este site tem como característica a possibilidade de um usuário consultar questões, gerar provas virtuais, fazer busca por palavra-chave, em um banco de dados armazenado no servidor.

Sendo o desenvolvimento prático de um site o objetivo final do trabalho bem como a simulação de um servidor Web, iniciou-se o processo de definição dos tipos de ferramentas e a escolha entre as opções disponíveis. Seguiu-se então o estudo das ferramentas escolhidas: sistema operacional Linux, banco de dados MySQL, servidor Apache Web Server e a linguagem de programação PHP. Após concluído o estudo, começou a criação do servidor Web com a instalação e configuração em forma padrão das ferramentas acima citadas em um computador pessoal. Com o servidor Web devidamente configurado, vieram os testes do funcionamento da linguagem de programação PHP para acesso à base de dados MySQL. O próximo passo foi a implementação do site baseada na construção da base de dados, definição da estrutura do site, implementação dos *scripts* PHP que geram as páginas HTML através das consultas ao banco de dados, e também as páginas HTML simples.

Com esse estudo teórico e prático, observou-se a grande utilidade de bancos de dados em um Web Site. Pois permite a implementação de várias soluções que dinamizam o conteúdo de um site, facilitam sua administração, possibilita a criação de aplicações cliente-servidor completas, comércio eletrônico e todas as facilidades que uma base de dados pode oferecer. Mesmo sendo uma solução gratuita, mostrou-se bastante eficiente e capaz de ser solução para quase todos os tipos de Web Site.

Palavras-chaves: Internet, PHP, MySQL, Linux, Apache, Site.

Abstract

The present work has as its primordial objective to define a set free tools for the Linux platform that can cover all the stages of development of a Web Site with the manipulation of a database. It was initially started the study of which tools were necerrary and available for the installation of a Web Server. From that, a Site to simulate college entrance exams was implemented. This site has as its characteristics the possibility of a user to consult tasks, generate virtual tests and search for keywords in a database loaded in the server.

Being the practical development of a site the final objective of the work, as well as the simulation of a Web Server, the process to define the kind of tools and the choice among the available options was started. Following that, the study of the chosen tools: Linux operational system, MySQL database, Apache Web Server and PHP programming language. After the conclusion of the study, the creation of the Web Server with the installation and configuration in pattern form of the tools aforementioned in a personal computer. With the Web Server appropriately configured, the tests for the functioning of PHP programming language were made for the access to MySQL database. The next step was the implementation of the site based in the construction of the database, definition of the site structure, implementation of PHP scripts that generate HTML pages, through consultation of the database and simple HTML pages as well.

It was observed the great utility of database in a Web Site with this theoretical and pratical study, as they allow the implementation of several solutions that make the content of a site dynamic, make its management easier, make possible the creation of complete client-server applications, e-commerce and all the facilities that a database can ofter. Even being a free solution, it has shown to be quite efficient and capable to be the solution for almost all kinds of Web Sites.

Keywords: Internet, PHP, MySQL, Linux Apache, Site.

Lista de Abreviaturas

API – Interface de programação de aplicativos

ASP – Active Server Page

CGI – Common Gateway Interface

FTP – File Transfer Protocol

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

HTTPs – Hyper Text Transfer Protocol Secure

NCSA – Centro Nacional de Aplicações para Supercomputação

PHP – Personal Home Page

PHP/FI – Personal Home Page/Form Interpret

RPM – RedHat Package Manager

SQL – Structured Query Language

SSL – Secure Socket Layer

URL – Uniform Resource Locator

WWW – World Wide Web

Índice de Figuras

FIGURA 3.1 - EXEMPLO DE CÓDIGO PHP EMBUTIDO EM UM CÓDIGO HTML. 30
FIGURA 3.2 - EXEMPLO DE ALTERNÂNCIA ENTRE OS MODOS PHP E HTML 31
FIGURA 3.3 - PASSOS PARA INSTALAÇÃO DO PHP EM SISTEMAS UNIX 33
FIGURA 3.4 - EXEMPLO DE COMENTÁRIO DE UMA LINHA EM PHP 38
FIGURA 3.5 - EXEMPLO DE COMENTÁRIO DE MAIS DE UMA LINHA EM PHP 38
FIGURA 4.1 - GRÁFICO COMPARATIVO DA UTILIZAÇÃO DO APACHE NO MUNDO 43
FIGURA 4.2 - COMPARAÇÃO ENTRE PHP COMO UM CGI E UM MÓDULO APACHE 44
FIGURA 5.1 - FLUXOGRAMA DA SEQÜÊNCIA DE AÇÕES EXECUTADAS DURANTE O PROCESSO DE VISITA DE UM USUÁRIO À HOME PAGE IMPLEMENTADA 49
FIGURA 5.2 - MODELO E-R. 54

Índice de Tabelas

TABELA 2.1 - POLÍTICA DE USO DO MYSQL NA WEB 20
TABELA 2.2 - TIPOS DE VARIÁVEIS SUPORTADAS PELO MYSQL 23
TABELA 2.3 - TABELA USER DO MYSQL 25
TABELA 2.4 - TAREFAS SENDO EXECUTADAS NO SERVIDOR 28
TABELA 3.1 - HABILITAÇÃO DO BANCO DE DADOS DURANTE A INSTALAÇÃO DO PHP. 34
TABELA 3.2 - EXEMPLOS DE CONFIGURAÇÃO DO PHP 36
TABELA 5.1 - DESCRIÇÃO DOS CAMPOS DA TABELA QUESTÕES. 54
TABELA 5.2 - DESCRIÇÃO DOS CAMPOS DA TABELA RESPOSTA. 54
TABELA 5.3 - DICIONÁRIO DE DADOS DA TABELA QUESTÕES. 55
TABELA 5.4 - DICIONÁRIO DE DADOS DA TABELA RESPOSTA. 55

Introdução

A Internet, a grande rede mundial de computadores, é hoje a maior força propulsora em quase todos os segmentos da informática, uma revolução nos meios de informação sem limites e em plena expansão.

Nos dias de hoje é possível compartilhar informações de todos os tipos , desde simples textos a imagens tridimensionais, de sons a vídeos de última geração com um simples clique, seja do mais remoto lugar aos grandes centros, tudo isso graças a Internet.

Atualmente esse processo de compartilhamento de informações vem criando novas formas de negociação, remodelando completamente o sistema de logística das empresas, estreitando ligações comerciais e pessoais. Todas essas informações e interação entre as partes envolvidas pode-se dar através de bancos de dados, independente de plataforma de hardware, sistema operacional, topologia de rede, aplicativo ou localização geográfica. Pode-se por exemplo estreitar a ligação de um representante de uma industria, fazendo um pedido via internet, inserindo os dados do pedido automaticamente no banco de dados da empresa, disponibilizando informações para a área de produção e esta para a expedição, transferindo parte da cadeia de produção para a internet.

Esse processo de utilização dos conceitos cliente-servidor, só era possível dentro de um espaço físico pequeno, ou com recursos tecnológicos avançados e caros. Hoje desenvolvem-se aplicativos inteiros que utilizam a Web como interface para acesso a banco de dados. Dentre os inúmeros sistemas pode-se citar o PHP, o ASP e o Cold Fusion.

Tendo em vista a crescente evolução destas ferramentas e sobretudo a sua especialização, optou-se neste trabalho por um segmento que ainda possui muito a ser desenvolvido que é o software gratuito. Assim, buscou-se identificar as partes componentes do sistema, que são:

a) Sistema Operacional - Linux
b) Servidor - Apache Web Server
c) Banco de Dados - MySQL
d) Linguagem de Programação - PHP

A motivação inicial para a escolha das ferramentas acima citadas foi o custo, que é nulo. Percebeu-se com o desenvolvimento uma série de outras facilidades que vieram a corroborar a idéia inicial que se tinha.

Com o objetivo de exemplificar a utilização das ferramentas, desenvolveu-se um ambiente de servidor de páginas Web. Nesse ambiente, implementou-se um site que permite ao usuário realizar provas virtuais de concurso vestibular, bem como consultas às questões cadastradas em um banco de dados.

Assim, o presente trabalho busca apresentar uma solução eficiente, completa e barata para um servidor Web que interprete *scripts* PHP que acessam o servidor MySQL. Para este fim, foi dividido da seguinte maneira.

No primeiro capítulo é apresentado o sistema operacional utilizado, o Linux. Incluindo um breve histórico, as distribuições existentes e algumas dicas de instalação.

O servidor de banco de dados MySQL está no segundo capítulo do trabalho, que contém suas principais características, as bases da instalação, sua utilização na Web, suas ferramentas clientes e alguns conceitos da linguagem SQL.

O terceiro capítulo do trabalho trata da linguagem de programação PHP, conceituando-o, apresentando os processos de instalação e configuração, descrevendo uma idéia básica da sintaxe da linguagem e citando alguns exemplos.

No quarto capítulo discorre o servidor Apache Web Server, mostrando o funcionamento básico de um servidor Web, os processos de instalação e configuração básica e descrevendo também sobre o protocolo SSL.

No quinto e último capítulo está documentada a implementação realizada neste trabalho. Abordando todo o seu processo de desenvolvimento desde a sua esquematização, passando pela modelagem dos dados, e chegando até a codificação em PHP e HTML.

1. Linux

Neste capítulo será abordado o sistema operacional Linux, iniciando com um breve histórico, passando pela descrição e comentários de algumas das principais distribuições e finalizando com algumas dicas e características do processo de instalação do mesmo.

1.1. Um breve histórico do Linux

"O sistema operacional Linux é uma versão gratuita distribuida do primeiro UNIX desenvolvido por Linus Torvalds na Universidade de Helsinque na Finlândia. O UNIX é um dos sistemas operacionais mais populares do mundo por causa de sua grande base de suporte e distribuição" [JAM 99]. O desenvolvimento do Linux contou com a participação de vários programadores UNIX e especialistas em Internet, dando a qualquer um que possuísse algum conhecimento e experiência necessários a possibilidade de desenvolver e até mesmo alterar o sistema.

Foi inicialmente desenvolvido como um passatempo para Linus. Foi inspirado no Minix, um pequeno sistema UNIX desenvolvido por Andy Tannenbaum.

Em agosto de 1991 a versão 0.01 ficou pronta, mas nenhuma publicação foi feita para esta versão, suas fontes não foram nem sequer executadas.

Dois meses depois, em 5 de outubro de 1991, Linus lançou a primeira versão oficial do Linux, a versão 0.02, porém pouco acrescentava em relação à versão anterior. Depois da versão 0.03, Linus desenvolveu o sistema até a versão 0.10, quando mais pessoas começaram a trabalhar no desenvolvimento. Dando seguimento ao sistema, Linus elevou o número da versão para 0.95, e segundo ele, logo estaria pronto para o lançamento de uma versão oficial.

Duas importantes fontes de software para o desenvolvimento do Linux foram o projeto GNU da *Free Software Fundation* (Fundação de Software Livre), uma organização de porte internacional que suporta a apóia projetos diversos ligados ao código livre, e os trabalhos em curso na Universidade de Berkeley. Destas duas fontes saíram diversos dos atuais utilitários e aplicativos do Linux, como o compilador C e os servidores de rede, entre outros.

Após passar por várias versões, o Linux é hoje um clone do UNIX completo. Possuindo compatibilidade com praticamente todos os grandes pacotes gratuitos de software. As versões disponíveis estão em vários idiomas, possuindo as variadas aplicações, processadores de texto, suportando redes e gerenciadores de banco de dados, e outras.

1.2. Distribuições do Linux

Sendo o Linux um software gratuito, a sua distribuição não é controlada por nenhuma organização ou entidade. Por isso, qualquer cidadão está apto à distribui-lo. Mas existem algumas empresas que cobram pela mídia.

Isso ocorre porque uma empresa pode obter gratuitamente o núcleo do Linux, trabalhar em cima desse núcleo, adicionar interfaces gráficas, aplicações, documentar e vender este pacote. Assim foi o surgimento das versões comerciais.

A característica de ser um software gratuito onde qualquer pessoa pode manipulá-lo, torna o Linux um sistema operacional de muitas distribuições. As distribuições disponíveis podem variar das mais completas, que vêm com todo o software que você precisa e mais alguns aplicativos, até as mais "enxutas", que são indicadas para usuários que não dispõem de muito espaço em disco. Essas versões "enxutas" possuem apenas o básico necessário para o seu funcionamento.

A seguir serão apresentadas e comentadas algumas distribuições do Linux disponíveis atualmente:

a) Red Hat: Esta distribuição tem anexada ao núcleo do sistema recursos de programas de configuração de recursos de sistema, interfaces gráficas, especializações para cliente e servidor, entre outras. A característica forte dessa distribuição é a facilidade de instalação. "O processo de instalação é feito através do pacote RPM (Red Hat Package Manager), com ele toda a estrutura de funcionamento do software é transferida para o disco, nos locais apropriados, facilitando a atualização do sistema. Junto com a distribuição Red Hat são fornecidas as interfaces gráficas X-Windows e KDE" [JAM 99].
b) Slackware: É uma versão que apesar de não suportar os pacotes da Red Hat, traz consigo um conversor que permite trabalhar com eles. O Slackware possui opções de segurança, como por exemplo, o de avisar que uma função de inicialização do sistema (*script*) foi alterada e que isto poderá trazer problemas. Possui a interface gráfica X-Windows e o gerenciador de janelas fvwm95.
c) Caldera: É uma versão voltada para uso empresarial. Possui uma série de drivers, ferramentas e recursos para interligação do Linux em rede, como servidor. Esta versão vem com uma licença de um módulo de desenvolvimento do banco de dados Sybase e o ambiente gráfico KDE, incorpora também compatibilidade com o mecanismo RPM da Red Hat.
d) Debian: É a versão indicada para programadores mais experientes. Possui a sua própria estrutura de pacotes, o formato DEB, que é utilizado para instalação do sistema e dos

aplicativos. Os especialistas o consideram a ferramenta de instalação mais versátil do mercado.

e) Conectiva: A distribuição Conectiva é um trabalho feito em cima da versão Red Hat, incluindo traduções de mensagens e textos de apoio para o português e definições de hardware que normalmente são usados no Brasil, como placas de controle, monitores, impressoras, mouses, etc.

1.3. Instalação do Linux

O objetivo de se instalar um sistema operacional é a de possibilitar dar inicio ao uso do computador. Instalar um sistema é transferir para uma unidade de disco uma parte deste sistema.

A instalação do Linux, por ele ser um sistema complexo, abrangente e escrito por diversas pessoas, não é considerada umas das tarefas mais simples. Ele difere-se do sistema operacional Windows, por exemplo, por possuir diversas formas de instalação, isso ocorre devido à duas peculiaridades do Linux:

a) Possui variadas fontes de instalação, podendo ser instalado a partir de um ou dois CDs, utilizando-se ou não um disquete de inicialização. Uma outra forma é via Internet ou via FTP (File Transfer Protocol). Ou ainda poderá ter seu conteúdo de instalação copiado para uma unidade de disco de um computador e a partir dali ser instalado.
b) Possui diferentes distribuições. Como cada distribuição tem suas características e finalidades, o processo de instalação de cada uma delas poderá ter seqüência, duração e procedimentos diversos.

A seguir serão descritos alguns recursos mínimos de máquina necessários para a instalação do Linux, e algumas características de dispositivos que seria importante o usuário ter conhecimento antes de iniciar o processo de instalação:

a) Processador: Processadores da Intel igual ou superior aos 386, processadores não-Intel como: Alpha, SPARC, Power PC, são suportados pelo Linux.
b) Memória: O Linux requer no mínimo 2 MegaBytes de memória, no caso de servidores de rede recomenda-se no mínimo 16 MegaBytes.
c) Discos e Controladoras: O Linux requer uma configuração especial para computadores que possuam controladoras SCSI, caso contrário, com discos normais será necessário criar uma partição para a instalação. Isso significa que o Linux poderá ser instalado no mesmo disco do DOS ou Windows, só que em partição diferente. O espaço da partição recomendável dependerá da quantidade de pacotes que serão instalados, a versão mínima do Linux requer

pelo menos 20 MegaBytes, mas instalando-se aplicativos, editores de texto, ambientes gráficos esse valor poderá chegar a 200 MegaBytes.

d) Monitor de Vídeo: A configuração do monitor de vídeo requer o conhecimento de que placa de vídeo e monitor estão instalados no computador. Além de modelo e fabricante, deve-se também passar como parâmetro no momento da instalação a quantidade de memória de vídeo que se possui, freqüências usadas pelo monitor, tipos de recursos, etc.

2. MySQL

2.1. O que é MySQL?

MySQL é um banco de dados relacional, desenvolvido para plataformas Linux – like, OS/2, Windows [YAR 99]. Sendo um software de livre distribuição para plataformas não-Windows que o utilizam em um servidor Web.

MySQL é um servidor multiusuário, multitarefa, compatível com o padrão SQL (Structured Query language – Linguagem de Consulta estruturada), linguagem essa amplamente utilizada para manipulação de dados em RDBMS (Banco de dados Relacionais), sendo considerada um ferramenta de manipulação de base de dados de tamanho moderado.

As principais características que destacam MySQL são: sua velocidade proporcionada pela sua implementação leve que não inclui na totalidade o suporte as instruções SQL; sua natureza de distribuição gratuita; facilidade de integração com servidor Web e linguagens de programação de desenvolvimento de sites dinâmicos, especialmente a linguagem PHP.

2.2. Breve Histórico

"O MySQL foi criado por Michael Widenius na companhia suíça TcX. Por volta de 1979 Michael desenvolveu um banco de dados chamado UNIREG, sendo rescritos em várias linguagens desde então" [YAR 99]. Em 1994, a empresa TcX começou o desenvolvimento de aplicações baseadas na Web, tendo como base o banco UNIREG, porém esse banco possuía muito "overhead" para obter sucesso em uma aplicação para geração de páginas dinâmicas na Web. Então a empresa TcX começou a procurar por outro banco o mSQL, uma ferramenta baseada em SQL mas com características pobres não possuindo por exemplo suporte a índices, e com desempenho inferior ao UNIREG.

Foi então que o desenvolvedor do banco UNIREG contatou o David Hughes criador do mSQL, para saber do interesse dele em unir os dois bancos. Sendo positivo o interesse de David , a empresa TcX resolveu desenvolver um novo banco, mas mantendo ao máximo a compatibilidade com mSQL. TcX foi esperta o suficiente para não reinventar o que já estava bem feito, ela construiu seu servidor baseado na estrutura que já estava montada do UNIREG e utilizou grande número de utilitários escritas para mSQL e fez API's para o novo servidor praticamente iguais ao mSQL. Como resultado usuários do mSQL que decidissem mudar para o novo servidor da TcX, teriam apenas que fazer pequenas e simples mudanças nos códigos existentes.

Então foi me maio de 1995 que, definitivamente, a primeira versão do MySQL foi lançada. Um dos parceiros da TcX sugeriu a distribuição do servidor na Internet, o objetivo disso era a utilização de um modelo pioneiro desenvolvido por Aladdin Peter Deutsch. O resultado foi um maior flexibilidade em sem "*copyright*", que fez do MySQL mais difundido gratuitamente do que mSQL.

2.3. Por que MySQL?

Sendo a concepção inicial do trabalho a utilização de ferramentas de livre distribuição para plataforma Linux-GNU, e o desenvolvimento de uma aplicação de banco de dados utilizando a Web como interface, fez-se necessário a escolha de um banco de dados que permitisse explorar as características básicas para implementação de uma aplicação cliente-servidor.

Entre as possibilidades encontradas surgiram três bancos de dados: MySQL, PostgreSQL e Interbase. O servidor PostgreSQL destacou-se por suas características de banco de dados objeto-relacional, permitindo explorar todas as possibilidades dos bancos relacionais, porém estendendo funções como classes, herança.

No processos de instalação foram encontradas dificuldades no que diz respeito a integração com o servidor Web, não sendo possível o acesso pela linguagem PHP.

O MySQL foi o banco de dados escolhido por apresentar extensa documentação, atualmente mais de 10 livros, milhares de sites na internet, mas principalmente pela sua fácil instalação e integração com o servidor Web.

Sua instalação através de RPM (RedHat Package Manager – Gerenciados de pacotes RedHat), é um processo simplificado, sendo criado toda a estrutura interna de arquivos no sistema operacional, bem como execução de *scripts* de inicialização e ajustes em arquivos de inicialização no servidor Web e módulo PHP.

2.4. Principais características

a) MySQL é um banco de dados multiprocessado, significando que pode utilizar vários processadores ao mesmo tempo.
b) Possui API's para C, C++, Java, Perl, PHP, Phyton e TCL.
c) Foi desenvolvido para várias plataformas incluindo ambientes Unix, OS/2 e Windows.

d) Permite operações e funções nas cláusulas *select* e *where*, bem como suporte as funções SQL(group by , order by), além de funções de grupo como: Count(), avg(), sum(), std(), max(), min().
e) Permite a seleção de diferentes tabelas de diferentes bases de dados em uma mesma *query*.
f) Suas características de privilégio de *password* são bastante flexíveis, permitindo inclusive a validação por "host".
g) Possui algoritmos de criptografia de *password*, fornecendo assim segurança aos dados gravados nas tabelas.
h) Permite a utilização de até 16 índices por tabela.
i) Capacidade para manipular bancos com até 50 milhões de registros.
j) MySQL foi escrito em C e C++
k) Permite conexões via TCP/IP
l) Permite acesso via ODBC.
m) Possui instruções para extração de informações relativas a tabelas, bancos, índices.

2.5. Política de uso na Web

A utilização de MySQL em sistemas operacionais não-Windows é gratuita se utilizada com um servidor WEB, não necessitando assim de licença para uso. Isso é válido mesmo que as aplicações Web que utilizem o banco sejam para fins comerciais, excluindo-se a comercialização do MySQL.

O suporte para MySQL pode ser contratado, no entanto para isso existem contratos de manutenção, sendo esses restritos as áreas de cobertura da equipe.

Apesar de ser distribuído livremente para uso na Web e em outras várias situações, que não cabem aqui serem mencionadas, existem vários tipos de utilização no qual se faz necessário a aquisição de licenças de uso, para citar alguns exemplos de valores, segue tabela 2.1 .

Nr. de licenças	Valor por cópia
1	U$ 200
10	U$ 150
50	U$ 120
100-999	U$ 40
1000-2499	U$ 25

Tabela 2.1 - Política de uso do MySQL na Web

Valores extraídos da documentação oficial – Fev/2000

2.6. Informações básicas sobre instalação.

O MySQL possui diversas versões, possuindo dois métodos de instalação: um através da compilação da distribuição dos fontes, a outra através da instalação dos binários.

No desenvolvimento deste trabalho foi utilizada a versão 3.22.25 de distribuição binária. Normalmente o MySQL possui um versão *beta*, a última a estar disponível, porém nem sempre é a versão mais estável, sendo assim, a melhor escolha deve ser sempre a da última versão estável.

Para obter um cópia do servidor MySQL é possível utilizar vários sites ao redor do mundo, que possuem as versões disponíveis para *download*, ou a utilização dos pacotes que acompanham a maioria das distribuições Linux.

Neste trabalho foi explorada apenas o processo de instalação sobre distribuição binária com a utilização de pacotes RPM(RedHat Package Manager). O processo de instalação para plataformas Linux, baseadas na distribuição RedHat, incluindo a distribuição brasileira Conectiva, é um processo bastante simplificado, pois todo o trabalho de criação de diretórios, execução de *scripts* de instalação, configuração de arquivos, reinicialização de processos, criação de links simbólicos, entre outros ficam invisíveis durante a instalação.

A utilização de pacotes possui como principal vantagem essa simplificação na instalação, por outro lado, por utilizar processos bastante automatizados, não permite que a instalação seja customizadas por usuários mais experientes e que necessitem adequar a instalação em seu sistema.

Os pacotes que devem ser utilizados na instalação são os que seguem abaixo:

a) Principal: MySQL-versão.i386.rpm . Este pacote possui o próprio servidor de Banco de Dados MySQL, deve ser instalado.
b) Programas Cliente Standart: MySQL-Cliente-versão.i386.rpm. Este pacote possui as ferramentas básicas para gerenciamento, criação e manutenção dos bancos criados.
c) Teste e Benchmark: MySQL-bench-versão.i386.rpm. Esse pacote permite que se façam testes de performance das estruturas criadas no Banco.

Em uma *shell* de um Linux baseado em RedHat, com RPM instalado e os pacotes já copiados, executa-se os comandos abaixo:

shell> rpm –i MySQL-versao.i386.rpm MySQL-Client-i386.rpm <enter>

Esse processo instalará o servidor e as ferramentas clientes básicas, permitindo assim seu funcionamento e criação dos bancos de dados. Durante esse processo o banco de dados será copiado com as permissões padrões. Ocorrerá a criação de entradas em *sbin/rc.d* para que se possa iniciar o servidor automaticamente, bem como os arquivos de executáveis e extensões dos aplicativos para os diretórios padrão.

Caso o processo ocorra sem erros já é possível testar instalação executando em uma shell o comando **mysql,** o qual iniciará a principal ferramenta cliente para interpretação de comandos SQL.

OBS: A Instalação através dos fontes do MySQL, requer algumas ferramentas no seu ambiente operacional: um compilador C++ compatível com ANSI, um programa make, um descompactador, entre outros. De forma simplificada será necessário compilar os fontes, e passar alguns parâmetros de instalação. Esse processo exige um conhecimento mais aprofundado das caraterísticas as ferramentas envolvidas.

2.7. Compatibilidade do Banco MySQL com a linguagem SQL

O banco de dados MySQL é compatível com boa parte dos comandos da padronização SQL 92, possuindo um conjunto não muito extenso de instruções porém com as funções mais utilizadas [YAR 99]. Além disso, possui extensões com funções bastante úteis, entre elas o *auto_increment* que permite a geração de números seqüências para registros em cada nova inserção executada em uma tabela. Alguns comandos próprios do MySQL não funcionarão com outros bancos, mas podem ser comentados de forma que durante um processo de importação sejam ignorados, mantendo assim a portabilidade de suas instruções.

Também não suporta a utilização de *subselects,* ficando assim deficiente em algumas situações de consulta. Não tem suporte também a *commit* e *rollback* conforme será discutido no próximo capítulo.

O MySQL suporta uma grande extensão de tipos numéricos, caracteres, caracteres de tamanho variável, tipos enumerados, sendo os mais comuns e mais compatíveis os que seguem a seguir:

Tipo	Descrição
Int	Valor inteiro
Real	Valor de ponto flutuante
Char(tamanho)	Caracter de tamanho fixo
Date	Campo de data padrão
Time	Campo de tempo padrão
Text(tamanho)	Caracter de tamanho variável

Tabela 2.2 - Tipos de variáveis suportadas pelo MySQL

2.8. Transações

Transações não são suportadas, até sua última versão. O objetivo segundo a equipe desenvolvedora é a implementação de suporte a transações atômicas, o que seria equivalente a transações sem a utilização de *rollback*(recurso para desfazer alterações). O finalidade de operações atômicas é permitir que se execute um grupo de instruções SQL, por exemplo vários comandos de atualização (UPDATE) e que haja garantia que nenhum outro processo interfira nos valores, tornado os dados inconsistentes.

A utilização de transações em bancos de dados permite garantir a integridade e a consistência das informações, e tem como característica a utilização de *Commit* e *RollBack,* ou seja, pode-se executar um conjunto de instruções e ter a garantia que todas elas serão executadas (Commit), ou no caso de alguma falha de *Hardware* ou de *Software* as alterações são desfeitas e retornam ao seu estado inicial antes das alterações(RollBack).

Por não utilizar esse conceito de transação em sua estrutura, o banco MySQL consegue uma grande melhoria no seu desempenho, por manter um conjunto de instruções menos complexas, sendo atualmente um dos bancos de dados mais rápidos para consultas.

Sendo assim, a sua utilização deve ser muito criteriosa, pois ao envolver processos de manipulação de informações que não podem ficar inconsistentes de forma alguma, o banco não oferece os recursos de segurança *commit* e *rollback*. Para contornar esse problema, utiliza-se o conceito de travamento e destravamento de tabelas.

O processo de travamento e destravamento de tabelas visa garantir a consistência da informação lida, porém o travamento deve ser invocado explicitamente na programação exigindo um controle rigoroso na programação. A seguir segue um exemplo do processo de travamento de tabela:

Instruções:

a) lock tables trans read, clientes write
b) select sum(value) from trans where cliente_id=10
c) update cliente set saldo = soma anterior where cliente_id = 10
d) unlock tables

Caso a tabelas não fossem travadas, existiria a possibilidade de outro processo inserir um novo registro na tabela *trans,* entre a execução do *select* e do *update* tornando assim a informação incorreta.

2.9. Store procedures e Triggers

O banco de dados MySQL suporta a utilização de *procedures* e *triggers*. A utilização de *procedures* possibilita que vários processos de programação sejam efetuados no próprio banco de dados, permitindo assim a diminuição do tráfego cliente-servidor bem como benefícios no que diz respeito a independência de aplicativos, pois os processos implementados no banco ficam transparentes para a aplicação que acessa os dados.

Pode-se definir *trigger* como sendo um ou vários procedimentos armazenados que são executados quando determinado evento ocorre, por exemplo, a cada vez que um registro é atualizado dispara-se a execução de certos procedimentos.

2.10. Segurança e privilégios de acesso

A função primária do sistema de privilégios do MySQL, é validar a conexão de um usuário de uma determinada estação e definir quais os privilégios o usuário tem com as instruções *Select*, *insert, update, delete* em um determinado banco de dados.

Além dessas instruções, são definidos quais os privilégios administrativos como criação de usuários, definição de permissões, possibilidade de iniciar e parar o servidor entre outras.

2.10.1. User names e password

O processo de acesso ao MySQL se faz através de nome de usuário e senha, sendo a senha opcional. Os nomes e senhas utilizados na autenticação não possuem qualquer ligação com os nomes utilizados e senhas utilizados no ambiente operacional (Linux).

Para que possa haver segurança no acesso aos dados do banco, deve-se utilizar a política de senhas para todos os usuários, caso contrário um usuário poderá acessar informações facilmente, além de facilitar a invasão ao banco por pessoas não autorizadas.

Algumas características importantes a serem citadas sobre a utilização de nomes e senhas:

a) Nomes de usuários podem conter até 16 caracteres alfanuméricos
b) Não existe ligação entre nomes do ambiente operacional e do banco.
c) MySQL criptografa o *password* usando um algoritmo diferente do que o usado em uma seção Linux.
d) Possui as funções **Password**(‘suasenha’) e **Encrypt**(‘suasenha’), sendo que a primeira utiliza um algoritmo do próprio MySQL e a segunda faz uma chamada ao sistema de criptografia do sistema Linux.

Todas as informações referentes à controle de acessos e privilégios ficam armazenas e um banco de dados dentro do MySQL chamado mysql , com as informações sobre usuários especificadas dentro de uma tabela com o nome de *user.* Esta tabela possui a seguinte estrutura de dados assim explicada:

Campo	**Tipo**	**Funçao**
Host	Char(60)	Host a qual se dará o acesso ao banco
User	Char(16)	Nome do usuário
Password	Char(16)	Senha do usuário (Campo pode ser criptografado)
Select_priv	Enum(‘n’,’y ‘)	Possibilidade de executar o comando select
Insert_priv	Enum(‘n’,’y ‘)	Possibilidade de executar o comando insert
Update_priv	Enum(‘n’,’y ‘)	Possibilidade de executar o comando update
Delete_priv	Enum(‘n’,’y ‘)	Possibilidade de executar o comando delet
Create_priv	Enum(‘n’,’y ‘)	Possibilidade criar banco de dados
Drop_priv	Enum(‘n’,’y ‘)	Possibilidade de apagar banco de dados
Reload_priv	Enum(‘n’,’y ‘)	Possibilidade de atualizar os privilégios definidos
Shutdown_priv	Enum(‘n’,’y ‘)	Possibilidade de finalizar o servidor
Process_priv	Enum(‘n’,’y ‘)	Possibilidade de gerenciar tarefas em execução
File_priv	Enum(‘n’,’y ‘)	Possibilidade de usar comandos que executam sobre arquivos
Grant_priv	Enum(‘n’,’y ‘)	Possibilidade de definir privilégios para outros usuários
Index_priv	Enum(‘n’,’y ‘)	Possibilidade de criar e apagar índices
Alter_priv	Enum(‘n’,’y ‘)	Possibilidade de executar o comando ALTER TABLE

Tabela 2.3 - Tabela user do MySQL

Outras tabelas completam o controle total de permissões e privilégios:

DB – esta tabela possui as informações de qual usuário pode executar quais comandos, de determinada máquina, exatamente sobre um banco de dados específico. Esta tabela se faz necessária, pois a tabela *user* não possui qualquer referência sobre quais bancos de dados eram definidos os privilégios, assim com a tabela *db* pode ter um controle mais específico.

Host – esta tabela tem por objetivo controlar o acesso a banco levando em consideração em qual máquina esta sendo efetuada a conexão, assim se a tabela *db* não identificar o *host* para um banco de dados e usuário, a tabela *host* define se o banco de dados em questão pode ser acessado de determinado *host*.

2.11. Como funciona a conexão ao MySQL

O processo de conexão com o banco requer que alguns parâmetros sejam passados ao servidor MySQL:

a) Hostname – Qual computador deseja efetuar a conexão.
b) usuário – O nome do usuário que deseja conectar
c) A senha – Senha do usuário que requisitou a conexão.

Esses dados são passados para o servidor através da aplicação cliente, por exemplo um página desenvolvida em PHP ou um módulo de administração.

No caso de omissão de algum parâmetro, a requisição de conexão assume os seguintes valores como padrão:

a) Hostname : LocalHost
b) Nome do usuário : Nome do usuário conectado no ambiente operacional
c) Password: Não passa nenhuma informação se omitido

Abaixo o exemplo de uma conexão através da linguagem PHP embutida em uma página HTML:

a) $conexão = mysql_conect(‘localhost’,’joao’,’xxxyyy’);
b) mysql_select_db(‘conexão’,’teste’);

Com a função nº 1 nativa da linguagem PHP, solicita-se uma conexão ao servidor MySQL, sendo o lugar da conexão o próprio computador(localhost), o usuário (joao) e a sua senha(xxxyyy). Caso executado com sucesso a função, a variável *$conexão* recebe a validação da conexão.

A segunda função seleciona qual o banco de dados que a conexão acima realizada deseja conectar, nesse caso foi conectado ao banco de dados *teste*.

2.12. Principais ferramentas para utilização do MySQL

2.12.1. MySQL

Talvez uma das mais importantes ferramentas para criação de banco de dados em MySQL seja a mysql, pois através dela cria-se uma conexão direta com o servidor MySQL, permitindo executar todos os comandos SQL disponíveis desde a criação de bancos até a definição de privilégios para o servidor. Com a execução do comando inicia-se a utilização do MySQL:

Shell> mysql –u joao –p teste

Com esse comando cria-se um conexão com o servidor MySQL para o usuário *joao* ao banco de dados *teste*, sendo solicitada sua senha.

Assim inicia-se a conexão comum *prompt* como esse: (mysql>) que permite a execução dos comandos SQL.

2.12.2. Mysqladmin

Mysqladmin é a ferramenta principal para administração de bancos de dados rodando em MySQL Server. Usando esta ferramenta pode-se criar, destruir e monitorar servidores de bancos de dados.

Para criar um banco dados pode-se utilizar esta ferramenta como segue exemplo:

Shell> mysqladmin – p create nomedobanco

A opção –p fala ao MySQL para solicitar ao usuário a senha de *root* que deve estar definida previamente. Caso o comando seja executado com sucesso, o servidor criará um novo banco, vazio e com o nome definido.

Para excluir um banco de dados utiliza-se o seguinte comando:

Shell > mysqladmin – p drop nomedobanco

Com esse comando apaga-se todos os arquivos referentes a banco de dados com o nome *nomedobanco*.

A ferramenta mysqladmin fornece um grande número de comandos que permitem monitorar o servidor MySQL. Um dos mais importantes é comando:

Shell > mysqladmin status

Através dele, pode-se analisar um grande variedade de informações sobre o Servidor, conforme abaixo:

a) Uptime – o tempo em que o servidor esta rodando
b) Threads – o número de tarefas que estão atualmente interagindo com o servidor. Pelo menos uma tarefa sempre estará sendo executada, é a tarefa que conta quantas outras tarefas estão sendo executadas
c) Questions – o número de *queries* que foram enviadas desde que o banco foi iniciado
d) Slow queries – o número de *queries* que levaram mais tempo do que o tempo padrão configurado.
e) Opens - o número de tabelas que foram abertas desde que o banco foi iniciado
f) Open tables – o número de tabelas atualmente abertas. Por ser multitarefa uma tabela pode estar aberta por mais de um processo, assim o número de tabelas abertas pode ser maior que o número de tabelas existentes no servidor.

Outro comando que contém informações sobre o servidor, mas de uma forma mais geral é o *mysqladmin version*, através dele pode-se saber: qual a versão do servidor, a versão do protocolo, o método pelo qual você está conectado ao servidor, por exemplo via *socket unix*, o nome do arquivo que está sendo usado para se comunicar com o servidor. Um exemplo de saída para este comando:

Server version	3.22.25
Protocol version	10
Connecton	estacao1 via Unix socket
Unix Socket	/tmp/mysql.sock
Uptime	50 min 10 sec

O comando *mysqladmin processlist* permite o monitoramento dos processos sendo executados.

Abaixo um exemplo de uma saída desse comando:

ID	User	Host	Db	Command	Time	State	Info
562	Joao	Cliente.com	Teste	Sleep	0		
985	Root	Localhost		Processes	0		

Tabela 2.4 - Tarefas sendo executadas no servidor

Esta tabela demonstra exatamente o que cada tarefa executada no servidor está fazendo.

ID : Possui o valor interno que identifica a tarefa executada, não possui relação com o número de processos do sistema operacional. É possível utilizar este número para finalizar a tarefa executando o comando *mysqladmin kill***.**

User: O usuário que está executando essa tarefa.

Host: O computador do qual o usuário está conectado.

Db : Qual o banco de dados o usuário está conectado.

Command: O tipo de comando que está sendo executado pela tarefa, podem ser :

a) Sleep – Tarefa esperando por entrada.
b) Quit – Tarefa sendo finalizada.
c) Init DB – Preparando-se para selecionar banco para conexão.
d) Query – Executando uma *query*, por serem comandos executados muito rapidamente, raramente aparecem na lista de processos.
e) CreateDB – Criando um novo banco de dados.
f) Drop – Apagando um banco de dados
g) Reload – Recarregando as tabelas com informações de acesso ao banco.
h) Shutdown – Tarefa finalizando todos as outras tarefas e parando o servidor
i) Statistics – Tarefa gerando estatísticas.
j) Processes – Tarefa examinado outras tarefas.
k) Connect - Negociando um conexão.
l) Kill – Tarefa finalizando outra tarefa

Como pode-se observar, o comando ***mysqladmin processlist*** permite fazer uma análise detalhada de todas as tarefas que estão sendo executadas, dando a possibilidade do administrador avaliar a necessidade de encerramento de tarefas em casos de falhas.

O processo de finalização do banco de dados se da através do comando:

```
Sheel> mysqldamin –p shutdown
```

3. Personal Home Page(PHP)

Neste capítulo serão apresentadas noções básicas da Linguagem PHP, incluindo uma breve definição, um histórico, algumas dicas de instalação e configuração em plataforma UNIX, além de noções sobre a sintaxe do mesmo e finalizando com exemplos.

3.1. O que é PHP ?

"PHP é uma linguagem que permite criar Web sites dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links. A diferença de PHP com relação a linguagens semelhantes a Javascript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas HTML(Hyper Text Markup Language) puro. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial" [BAR 99].

Para interagir com o Banco de Dados, é possível se inserir dentro do código HTML, além da linguagem de programação C, a linguagem para manipulação de Banco de Dados SQL (Structured Query Language).

Uma característica desta linguagem é a possibilidade de inserir o código PHP dentro do código HTML, o que não acontece em outros scripts CGI como o Perl ou o C, por exemplo.

Para simplificar tem-se o exemplo a seguir, onde a frase "Exemplo de código PHP" é simplesmente impressa na tela:

```
<html>
<body>
<?php
print "Exemplo de código PHP";
?>
</body>
</html>
```

Figura 3.1 - Exemplo de código PHP embutido em um código HTML.

O código PHP permite ainda que durante a execução haja uma alternância entre o seu modo e o modo HTML.

```
<html>
<body>
<?php
print "Modo PHP";
?>
<hr>
<center><b>Modo HTML</b></center>
</hr>
<?php
print "Modo PHP de novo";
?>
</body>
</html>
```

Figura 3.2 - Exemplo de alternância entre os modos PHP e HTML

O que o distingue de outros CGI cliente-servidor é que o código é executado no servidor. Se o servidor tem um script rodando, o cliente terá que receber o resultado da execução daquele script. Para isso o servidor deve estar configurado para processar todos os seus arquivos HTML com o PHP embutido nele.

Muito do PHP é uma combinação das Linguagens Perl, Java, C e C++. Sendo que a sintaxe é originada do C, tornado o seu aprendizado muito fácil até mesmo para programadores novatos. Possui ainda a capacidade de realizar sofisticados cálculos matemáticos, fornecer informações em rede, e muito mais. Atualmente está disponível para as plataformas Unix (FreeBSD, Linux, etc) e Windows (9x e NT).

3.2. Histórico

O desenvolvimento do PHP começou em meados de 1994 por Rasmus Lerdorf que lançou uma versão apenas para teste e demonstração, e utilizado pelo seu criador para o desenvolvimento de uma Home Page visando divulgar e aplicar testes práticos à nova ferramenta.

A primeira versão à ser usada por outros programadores só foi disponibilizada em 1995 e foi chamada de "Personal Home Page Tools". Essa versão foi considerada apenas como uma

revisão da anterior, tanto que não foi muito utilizada. Não foram feitos muitos trabalhos com ela, apenas alguns utilitários já bastante comuns para as Home Pages da época, um guestbook, um contador de visitas e algumas outras aplicações pequenas. Esta versão foi revisada e refeita em meados de 1995 e nomeada de PHP/FI 2.0. O FI era a indicação de um outro pacote que Rasmus havia criado para permitir a interpretação de comandos HTML, o Form Interpret. Ele combinou os *scripts* da Personal Home Page Tools com o Form Interpret e adicionou suporte ao mSQL, estava criado o PHP/FI. A partir daí o PHP/FI deu um grande passo em seu desenvolvimento, e pessoas alheias ao seu projeto começaram a contribuir em seu código fonte.

É difícil apontar estatísticas precisas, mas estimasse que em 1996 cerca de 15.000 sites em todo o mundo utilizavam o PHP/FI, já em meados de 1997 este número subiu para algo em torno de 50.000. Em 1997 houve outra grande mudança no PHP, o seu criador Rasmus Lerdorf anunciou que devido ao grande número de projetos que havia recebido de outras pessoas iria a partir dali montar um grande time para tratar do desenvolvimento e administração do PHP. Então iniciou-se o desenvolvimento de uma nova versão, o PHP 3.0. Nesta versão uma parte do código do PHP/FI foi portada no PHP 3.0 e a outra foi completamente rescrita [BAK 99].

Em 1999 estimava-se que o número de sites que utilizam o PHP tanto a versão PHP/FI 2.0 como a PHP 3.0, gira em torno de 1.000.000 em todo o mundo [BAK 99].

3.3. Bancos de Dados

“Conectar um Banco de Dados via Internet nunca foi uma das tarefas mais simples. Talvez a mais forte e mais significante característica do PHP em se tratando de acesso a Banco de Dados via Internet, é o vasto número de Banco de Dados à que ele dá suporte”.[FIS 99] Os Bancos de Dados a seguir são os suportados pelo PHP:

a) Adabas D
b) Interbase
c) Solid
d) DBase
e) MSQL
f) Sybase
g) Empress
h) MySQL
i) Velocis
j) FilePro

k) Oracle
l) Unix dbm
m) Informix
n) PostgreSQL

3.4. Instalação em Sistemas UNIX

Para instalar o PHP existem alguns pré-requisitos. A configuração inicial é:

a) Habilidade básica em Sistemas UNIX – ser capaz de operar o comando "make"
b) Um Compilador da Linguagem de Programação C
c) Um Servidor Web

Tendo essas ferramentas instaladas e possuindo o devido conhecimento para operá-las, o PHP já pode ser instalado. Por se tratar de uma Linguagem nativa do Apache Web Server, então primeiramente deve-se instalar o Servidor Web e logo em seguida o PHP como sendo seu módulo.

```
1. gunzip apache_1.3.x.tar.gz
2. tar xvf apache_1.3.x.tar
3. gunzip php-3.0.x.tar.gz
4. tar xvf php-3.0.x.tar
5. cd apache_1.3.x
6. ./configure –prefix=/www
7. cd ../php-3.0.x
8. ./configure –with-mysql –with-apache=../apache_1.3.x –enable-track-vars
9. make
10. make install
11. cd ../apache_1.3.x
12. ./configure –prefix=/www –activate-module=scr/modules/php3/libphp3.a
13. make
14. make install
```

Figura 3.3 - Passos para instalação do PHP em sistemas UNIX

Feita a instalação do Apache Web Server e do PHP respectivamente o próximo passo é tratar da configuração, mas isso será visto mais adiante. Agora mostra-se como se faz, para no momento da instalação dar suporte aos Bancos de Dados suportados pelo PHP.

No momento da instalação, na linha 8 da figura 3.3, o Banco de Dados MySQL foi configurado como o Banco de Dados que será suportado pelo PHP durante a sua execução. Mas como foi feito com o MySQL pode-se configurar outros Bancos de Dados também, e não apenas um mas quantos forem necessários, desde que, é claro, seja um Banco de Dados suportado. A maneira como isso será feito é a seguinte:

Bancos de Dados	**Diretivas**	**Descrição**
Adabas D	-with-adabas=DIR	Sendo que o parâmetro a ser passado é o diretório onde o Banco de Dados Adabas D será instalado, tendo como default é o diretório /usr/local/adabasd.
Dbase	-with-dbase	Suporte para Dbase, o qual é habilitado por default.
FilePro	-with-filepro	Suporte para Filepro, o qual é habilitado por default.
MSQL	-with-msql=DIR	Adiciona suporte ao mSQL. DIR corresponde ao diretório para instalação, sendo default o diretório /usr/local/Hughes.
MySQL	-with-mysql=DIR	Adiciona o MySQL com DIR sendo o diretório para instalação, tendo como default o diretório /usr/local.
Oracle	-with-oracle=DIR	Suporte para Oracle 7.0 até 7.3
PostgreSQL	-with-pgsql=DIR	O parâmetro é o diretório de instalação do PostgreSQL, tendo como default o diretório /usr/local/pgsql.
Solid	-with-solid=DIR	O diretório default para instalação é o , /usr/local/solid.
Sybase	-with-sybase=DIR	O parâmetro é o diretório de instalação do Sybase, tendo como default o diretório /home/sybase.
Sybase-CT	-with-sybase-ct=DIR	Adiciona suporte ao Sybase-CT. O diretório default para instalação é o /home/sybase.
Velocis	-with-velocis=DIR	O diretório default é o /usr/local/velocis.

Tabela 3.1 - Habilitação do Banco de Dados durante a instalação do PHP.

Existe uma outra forma de instalar o PHP, através dos pacotes RPMs (RedHat Package Manager). “Os pacotes RPMs são um recurso que contém todos os arquivos que são necessários para instalação em um único pacote, permitindo que tudo seja feito automaticamente. Isso evita a necessidade de descompactar e compilar arquivos como foi feito na instalação anterior” [JAM 99]. No caso do PHP, a instalação utilizando pacotes RPMs pode não ser muito recomendável, tendo em vista que o PHP possui algumas configurações que são específicas no código. Instalando o PHP dessa forma o Apache Web Server inicialmente não processará as páginas em formato PHP, sendo necessário alguns ajustes nos arquivos de configuração do Apache Web Server.

3.5. Configuração

A configuração do PHP é toda ela baseada em um arquivo, o php3.ini, que é o arquivo que será lido sempre que as páginas em PHP forem executadas no servidor. Ele possibilita a configuração de todos os diretórios e seus respectivos caminhos envolvidos na execução do PHP, bem como de todos os Bancos de Dados suportados, além da segurança e do limite de memória que será disponibilizada para a execução dos arquivos PHP. A seguir tem-se alguns exemplos de configuração:

Configuração	Descrição
doc_root string	É o diretório de root no servidor, em modo de segurança nenhum arquivo que estiver fora deste diretório ficará disponível para usuários comuns.
Include_path string	Especifica onde devem ser armazenados os arquivos em PHP para serem processados e exibidos.
user_dir string	Diretório destinado para uso dos usuários de arquivos PHP.
mysql.allow_persistent boolean	Define se é permitido insistência à conexão ao Banco de Dados.
mysql.max_persistent integer	Se for permitido, informar o máximo de insistências permitidas por processo.

mysql.max_links integer	O número máximo de conexões ao Banco de Dados permitidas por processo, incluindo as insistências.

Tabela 3.2 - Exemplos de configuração do PHP.

Mas, dependendo da forma como o PHP foi instalado será necessário mais algumas configurações. Se a instalação foi utilizando pacotes RPM então devemos também configurar o arquivo do Apache Web Server *httpd.conf.*

Assumindo que a instalação foi toda ela com RPM, algumas linhas do arquivo *httpd.conf* deverão ser descomentadas ou até mesmo adicionadas:

#Extra Modules
```
AddModule mod_php.c
AddModule mod_php3.c
AddModule mod_perl.c
```

#Extra Modules
```
LoadModule php_module modules/mod_php.so
LoadModule php3_module modules/libphp3.so
LoadModule perl_module modules/libperl.so
```

E deve-se adicionar a linha:
```
AddType applications/x-httpd-php3.php3
```

Feito isso o Apache Web Server está apto para executar arquivos em formato PHP.

3.6. Sintaxe Básica

A seguir será dada uma idéia geral a respeito da sintaxe do PHP.

3.6.1. Delimitadores

Existem quatro formas de se delimitar um conjunto de código PHP dentro de um código HTML.

O interpretador reconhece um código PHP pelas seguintes *tags*:

<?
[código]
?>

Esta é a forma mais utilizada.

<?php
[código]
?>
Forma também bastante comum e utilizada pelos programadores.

<script language=”php”>
[código]
</script>
Este caso é utilizado em editores HTML que não reconhecem o PHP como uma *tag*. Se as outras formas forem utilizadas o editor poderá acusar erro no HTML.

<%
[código]
%>
É uma forma que está sendo eliminada para futuras versões do PHP.

3.6.2. Separador de Instruções

Assim como nas linguagens de programação mais conhecidas, como o C e o Perl por exemplo, no PHP é necessário separar cada instrução do conjunto de código por ponto-e-vírgula.

3.6.3. Definição de Variáveis

Todas as variáveis são definidas através de um “$” seguido de uma string que deve iniciar por uma letra ou o caracter “_”.

O PHP é *case sensitive*, ou seja, diferencia maiúscula de minúscula, a variável $nome é diferente da variável $NOME. Como o PHP já possui algumas variáveis pré-definidas cujos nomes são formados por letras maiúsculas, é aconselhável evitar-se usar variáveis com letras maiúsculas.

3.6.4. Comentários

A forma de comentário do PHP é semelhante ao das linguagens C e C++ e com os comentários do UNIX shell.

Pode-se fazer comentários de duas formas:

a) Comentando uma linha:

É feito utilizando-se o caracter “#” ou duas barras (//).

```
<?php  Print “Correto”; #esta é uma forma  ?>
<?php  Print “Correto”; //esta é outra forma ?>
```

Figura 3.4 - Exemplo de comentário de uma linha em PHP

b) Comentando mais de uma linha:

É aplicado utilizando-se os caracteres “/*” para o início do comentário e “*/” para finalizar o mesmo.

```
<?php Print “Correta”; /* Comentando mais de
uma linha em PHP.
*/
```

Figura 3.5 - Exemplo de comentário de mais de uma linha em PHP

3.7. Exemplos

3.7.1. Um exemplo simples

Após a realização dos processos de instalação e configuração, é importante se fazer um exemplo para certificar-se de que tudo está correto. O script a seguir trata-se de um exemplo simples onde será produzida apenas uma saída.

```
<html>
<head><title>Testando o PHP</title></head>
<body>

<?php
print "Meu primeiro script em PHP";
?>

</body>
</html>
```

Exemplo de um script simples em PHP.

Feito isso deve-se salvar o arquivo, como “exemplo.php3” por exemplo, dentro do diretório dos documentos do Apache Web Server (html). Em seguida na janela do navegador digite o endereço: “http://localhost/exemplo.php3”. Se a página exibida no navegador contiver apenas a saída especificada no código PHP é sinal que tudo está funcionado corretamente. Exibindo-se o código fonte da página, teremos o seguinte:

```
<html>
<head><title>Testando o PHP</title></head>
<body>

Meu primeiro script em PHP

</body>
</html>
```

Exemplo – Código fonte do que será exibido no navegador.

Este código permite mostrar como o PHP funciona, o script foi executado no servidor deixando disponível ao usuário apenas o resultado em forma de código HTML.

3.7.2. Exemplo de PHP com formulários HTML

O próximo exemplo mostra como o PHP trabalha com formulários HTML. Sempre que se pressiona o “Submit” em um HTML alguma informação é enviada ao servidor para que se produza uma resposta que satisfaça a solicitação feita no formulário, para isso o PHP trata essas informações como variáveis.

```
<html>
<head><title>Testando o PHP</title></head>
<body>

<?php
if ($texto != "")
  print "Você digitou “$texto”<br><br>”;
?>
```

```
<form method=post action="<? print $PATH_INFO; ?>">
<input type="text" name="texto" value="" size=10>
<br>
<input type="submit" name="sub" value="Enviar!">
</form>

</body>
</html>
```

Exemplo de um PHP com formulário HTML.

Salvando esse arquivo e carregando-o no navegador ele inicialmente será apenas um formulário para digitar um texto. Digitando o texto pela primeira vez, a variável *$texto* deixará de ser vazia e conterá o texto digitado, então a partir daí o PHP exibirá antes do formulário o conteúdo da variável *$texto*.

3.7.3. Exemplo do PHP acessando uma base de dados

Para interagir com uma base de dados SQL existem três comandos básicos que devem ser utilizados: um que faz a conexão com o servidor de banco de dados, um que seleciona a base de dados a ser utilizada e um terceiro que executa uma *query* SQL.[BAR 99]

O banco de dados utilizado neste exemplo será o Banco de Dados MySQL, que pode ser copiado gratuitamente no endereço *http://www.mysql.org.*

a) Conectando ao servidor de banco de dados:

Para conectar-se ao servidor de banco de dados o comando é *mysql_connect*, a sua sintaxe é a seguinte:

Int mysql_connect(string [hostname[:port][:/path/to/socket]], string[usuário], string[senha]);

Abre uma conexão a um servidor MySQL, retorna *id_link* se tiver sucesso, caso contrário, retorna *False*.[FIS 99]

Os parâmetros exigidos neste comando são o endereço do servidor (hostname), o nome do usuário (usuário) e a senha para conectar-se ao servidor.

```
$conectou = mysql_connect(“localhost”, “root”, “pwdphp”);
```

Exemplo de conexão ao servidor de banco de dados.

b) Selecionado o Banco de Dados

Após feita a conexão deve-se selecionar qual banco de dados será utilizado, para isso usa-se o comando *mysql_select_db*, que possui a sintaxe:

Int mysql_select_db(string nome_bd, int [id_link]);

Os parâmetros são o nome do banco de dados a ser selecionado e um identificador de conexão.

```
$conecbd = mysql_select_db (“vestiba”, $conexao);
```

Exemplo de conexão a um banco de dados.

c) Executando *queries* em SQL no banco de dados

Tendo conectado ao servidor e selecionado o banco de dados, pode-se manipular praticamente todo o banco de dados utilizando-se SQL. Para se escrever comandos SQL no PHP utiliza-se o comando *mysql_query*, que possui a seguinte sintaxe:

Int mysql_query (string consulta, int [id_link]);

Envia uma consulta ao servidor MySQL, retorna *True* se tiver sucesso, caso contrário, retorna *False*.[FIS 99].

O exemplo a seguir demonstra a criação de uma tabela com SQL:

```
$Criatable = mysql_query(“CREATE TABLE nomes (codigo INT AUTO_INCREMENT PRIMARY KEY, nome CHAR(40), email CHAR(50))”, $conexao);
```

Exemplo de escrita SQL em um código PHP.

Assim como no exemplo acima foi executado o comando SQL *create table*, qualquer comando de escrita SQL poderia ter sido executado.

4. Apache Web Server

4.1. O que é um servidor Web

Um servidor Web é o responsável pelo controle de distribuição de páginas na Internet. Sua função é aguardar por requisições de páginas de um cliente (browser), e ao receber uma requisição ele retorna dados ao cliente, normalmente paginas HTML com imagens. Ao receber os dados enviados pelo servidor, o *browser* apresenta os dados para o usuário. Os servidores são em conceito muito simples, fazendo basicamente esse processos de resposta à requisições.

O processos de comunicação entre o cliente e o servidor são feitos através do protocolo de transferencia de Hyper Texto (HTTP - HyperText Transfer Protocol), protocolo esse que padroniza o modo de envio e recebimento de dados, tendo como característica principal a sua independência total quanto a plataforma, podendo por exemplo, estabelecer comunicação entre um servidor Linux, e um cliente *Macintosh* sem qualquer problema.

"As informações que trafegam pela Internet são na sua grande maioria feitos com a Linguagem de marcação de Hyper Texto (HTML - Hypertext Markup Language). Essa linguagem é a base principal de um ambiente Web, permitindo que exista essa grande gama de dados disponíveis para toda a Internet" [KAB 98].

Atualmente existem no mercado cerca de 500 servidores Web, disponíveis nas mais diversas plataformas, de distribuição gratuita até valores que chegam a milhares de dólares, dependendo dos recursos envolvidos na ferramenta [KAB 98]. Algumas das características principais a serem observados em um servidor Web são as seguintes: tipo de distribuição (gratuita ou paga), processo de instalação, facilidades de customização, performance e consumo de recursos, suporte técnico e módulos disponíveis são as principais informações a se levar em consideração.

4.2. O servidor Apache

O servidor Apache é uma das mais de 500 ferramentas disponíveis nesse segmento. Ele está disponível para as plataformas ,Unix Like, Solaris, Sun , Windows entre outras. Sendo um software de livre distribuição. Atualmente Apache é o servidor mais utilizado na Internet tendo aproximadamente 35% de todo o mercado, segundo a pesquisa do Site http://www.netcraft.com/survey/, vide gráfico comparativo (figura 4.1). Algumas das características que o destacam é o fácil processo de instalação que possui e uma configuração de arquivos bem funcional, permitindo que sejam feitos ajustes nas configurações, sem que seja

necessário reiniciar o servidor. O suporte à utilização de protocolos de transferência de arquivos com segurança, também tem outra característica importante implementada através de módulos no servidor.

O Apache é implementado como um conjunto de módulos, sendo grande partes desse módulos desenvolvidos por terceiros. Com a utilização desse módulos consegue-se uma grande flexibilidade na sua utilização, trazendo incontáveis funcionalidades para o servidor.

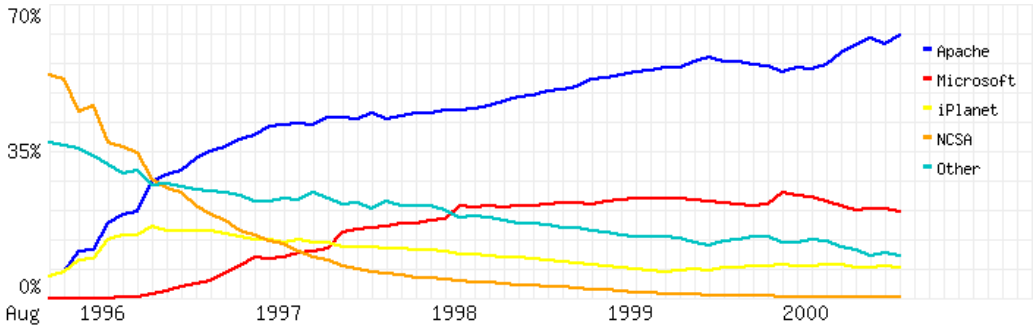

Figura 4.1 - Gráfico comparativo da utilização do Apache no mundo

4.3. Breve Histórico

Apache foi criado em 1995 quando um grupo de desenvolvedores chamados *The Apache Team* - Equipe Apache, consolidaram uma série de *patches* - remendos- que eles havia feito para o HTTP *daemon* do Centro Nacional de Aplicações para Supercomputação (NCSA). Um ano depois, Apache já era o mais popular servidor na Web.

4.4. Instalação

O processo de instalação do servidor Apache, no trabalho desenvolvido, visaza apenas criar um ambiente com configurações padrões, sendo apenas utilizadas configurações específicas para o uso do módulo PHP. Sendo assim, foi instalado um pacote RPM da distribuição brasileira da conectiva. Esse processo é bastante simplificado, não permitindo configurações específicas, porém sua instalação é bastante eficiente permitindo já ao seu término a inserção de arquivos HTML para serem processados.

Em um ambiente Linux, no diretório que contém o pacote Apache digita-se:

```
Shell>rpm –i apache-versão.i386.rpm
```

Com esse processo os módulos padrões, bem como as configurações, os *scripts*, os diretórios

default e os diretórios padrões serão configurados sem intervenção do usuário.

4.4.1. Os módulos do Apache

Módulo é um componente de software que adiciona funcionalidade para o servidor Apache. Através deles, define-se quais as características estendidas serão usadas no servidor. Um exemplo de definição de um módulo à ser usado, seria algo como:

```
Add module mod_php3.c
```

Esta linha no arquivo de configuração do apache, *httpd.conf*, permite a utilização do interpretador PHP caso este tenha sido instalado. Existe uma série de módulos que são instalados na configuração padrão, sendo que cada um dos módulos necessita mais memória e espaço do arquivo executácel apache. Assim, com a configuração dos arquivos pode-se escolher quais módulos serão usados permitindo assim melhor a rapidez do servidor e criar configurações de acordo com a necessidade do usuário.

4.4.1.1. O módulo PHP

O interpretador PHP pode ser compilado tanto como um interpretado CGI isolado do Apache ou como um módulo do Apache.

Se configurado como interpretador CGI, toda vez que um script PHP precisa ser interpretado o servidor Web lança uma instância do PHP, isso obviamente causará diminuição de performance, pois serão dois processos sendo executados, vide figura abaixo:

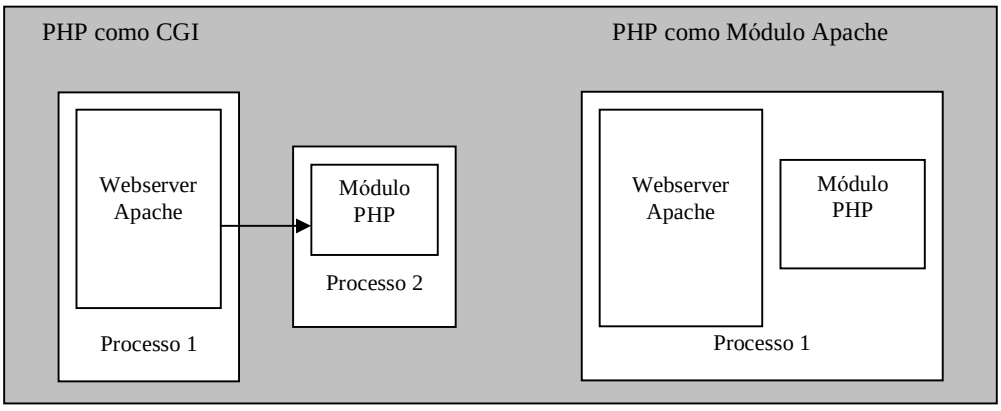

Figura 4.2 - Comparação entre PHP como um CGI e um Módulo Apache

"Quando compilado como módulo do Apache o PHP roda no mesmo espaço de memória do processo do servidor Apache, causando aumento da performance. Além disto, características

como conexões persistentes ao banco de dados e permissão de uma segurança adicional somente estão disponíveis no PHP como módulo" [CAS 00].

4.5. Configuração

4.5.1. Arquivos de configuração

Os três principais arquivos de configuração do servidor apache são:

a) Httpd.conf
b) Access.conf
c) Srm.conf

Estes três arquivos são textos puro, que são usados na inicialização do servidor para dizer como o servidor deve rodar. Neles existem dois tipos de informação: diretivas do servidor e comentários opcionais.

Uma diretiva é como um comando para o servidor, dizendo à ele qual tarefa deve executar.

4.5.1.1. Httpd.conf

Este tem dezenas de diretivas, que definem várias características de inicialização. Como por exemplo:

a) A porta Linux na qual o servidor receberá as requisições
b) Arquivos de log
c) Com que usuário o servidor estará rodando
d) E-mail do administrador
e) Diretórios de configuração
f) Erros e log
g) Nome do servidor
h) Tempo limite entre recebimento e envio de informação
i) Número máximo de requisições de páginas

Estas e outras diretivas definem a "espinha dorsal" do servidor.

4.5.1.2. Srm.conf

Este arquivo é o que define a configuração dos recursos disponíveis. Ele diz quais recursos serão oferecidos pelo Web Site, e quando e onde eles serão oferecidos. Algumas diretivas ficam

incluidas neste arquivo:

a) Diretório onde ficarão os documentos
b) Arquivo padrão de inicialização do diretório
c) Quais ícones devem ser mostrados para os diferentes tipos de arquivos
d) A linguagem que um documento pode usar, por exemplo o português
e) Os álias utilizados para os diretórios
f) Os diretórios dos scripts para utilização de CGI's
g) Mensagens customizadas de erros

A explicação destas e de outras diretivas contidas neste arquivo poderiam tomar várias páginas, porém de forma global observa-se que este arquivo define características de estrutura interna de diretórios, links simbólicos, definição de diretórios para usuários, imagens e mensagens de erros exibidas no *browser*, entre outros.

4.5.1.3. Access.conf

Este é usado para definir as permissões de acesso e ítens como: arquivos, diretórios e scripts do Web Site. Algumas diretivas importantes:

a) Cada diretório que o apache faz acesso pode ser configurado com respeito à quais características estão habilitadas ou desabilitadas
b) O nome dos diretórios e sub-diretórios que podem ter links simbólicos
c) Opções para arquivos padrões e diretório, por exemplo, se um dirtório não possue uma página índice pode-se definir neste arquivo qual página será exibida

4.6. Protocolo SSL (Secure Socket Layer)

Usar um servidor com suporte SSL é uma boa maneira de providenciar segurança para um Web Site sem a necessidade de nenhuma mudança nos códigos das páginas. O que o SSL faz é usar criptografia para proteger o fluxo de informação entre o servidor e o *browser*. Este protocolo não somente encriptografa a informação que trafega na Internet mas, também garante autencidade tanto do servidor quanto do cliente, ou seja, um usuário de um Site com o protocolo SSL tem certeza da origem dos dados.

O SSL usa uma técnica de encriptação chamada criptografia de chave-pública, onde o servidor envia uma chave-pública para o cliente criptografando informações na qual somente o servidor poderá descriptografar com uma chave-privada. O cliente usa a chave-pública para encriptografar e enviar ao servidor sua própria chave identificando ela somente para o servidor e

prevenindo um ataque no meio da comunicação dos dados.

HTTP seguro (HTTPs) se distingue de conexões regulares servindo informações numa porta diferente da 80, normalmente na porta 443. Clientes que solicitam uma URL com HTTPs automaticamente são direcionados para a porta 443, tornando fácil para o servidor responder as requisições para os diferentes protocolos.

Existem muitas soluções para implementar SSL com Apache, incluindo Apache-SSL e as implementações comerciais *StrongHold* e *Raven SSL*.

5. Implementação

O presente capítulo tem por objetivo apresentar e descrever o sistema implementado neste trabalho. A parte referente à modelagem dos dados para a construção da base de dados também está descrita neste capítulo incluindo descrição das tabelas, modelo e-r, dicionário de dados e a sintaxe SQL utilizada para a criação das tabelas no banco de dados.

As ferramentas utilizadas foram:

a) Sistema Operacional – Linux (Capítulo 1) – versão Conectiva 4.2
b) Banco de Dados – MySQL (Capítulo 2) – versão 3.22.25
c) Linguagem de Programação – PHP (Capítulo 3) – versão 3.0
d) Servidor Web – Apache Web Server (Capítulo 4) – versão 1.3.9

De posse e tendo o conhecimento necessário das ferramentas mencionadas acima, foi desenvolvida uma Home Page onde usuários podem realizar provas virtuais de concursos vestibulares onde, além disto, pode-se consultar as questões cadastradas no banco de dados.

A prova virtual possibilita que o usuário teste seus conhecimentos nas diversas matérias que compõem um concurso vestibular durante o processo de preparação do candidato. Essas matérias são: Matemática, Português, Física, História, Química, Geografia e Biologia. Para a realização da prova o usuário tem a possibilidade de escolher alguns critérios das questões que serão aplicadas na prova virtual. Por exemplo, além da matéria que se deseja, pode-se também classificar as questões pelo ano em que elas foram aplicadas e o local de onde elas fizeram parte, ou seja, em qual universidade ela foi aplicada. Outros critérios possíveis são o número de questões que se quer na prova e o nível de dificuldade da mesma, que pode ser fácil, intermediário, difícil ou aleatório. Com todos esses parâmetros passados já é possível realizar a prova virtual. Após a elaboração e realização da mesma, ou seja, depois que o usuário já tiver respondido todas as questões, é possível se fazer a correção retornando as respostas corretas confrontadas com as respostas dadas pelo usuário.

Para a realização da consulta ao banco de dados, igualmente à prova, deve-se passar parâmetros, que são matéria, local e ano. Um diferencial é a possibilidade de se consultar por texto, isso significa que através de uma palavra-chave digitada num campo disponível, tem-se retornado um resultado de consulta com questões referentes àquela palavra-chave. Isto é possível passando-se ou não os outros parâmetros, ou seja, pode-se realizar somente a consulta por palavra-chave.

Com a apresentação de uma breve descrição da implementação, onde deu-se uma idéia do funcionamento e do objetivo da mesma, o capítulo segue com uma descrição detalhada do processo de construção da Home Page para explicação da implementação de todos os itens que aqui foram citados. Primeiramente será apresentado um fluxograma com a descrição dos seus módulos para compreensão mais aprofundada, em seguida todo o processo de modelagem dos dados passando para as funções PHP que foram utilizadas e finalizando com a implementação de todos os arquivos PHP e HTML que compõem a Home Page.

5.1. Fluxograma

A figura 5.1 representa o fluxograma do sistema implementado neste trabalho.

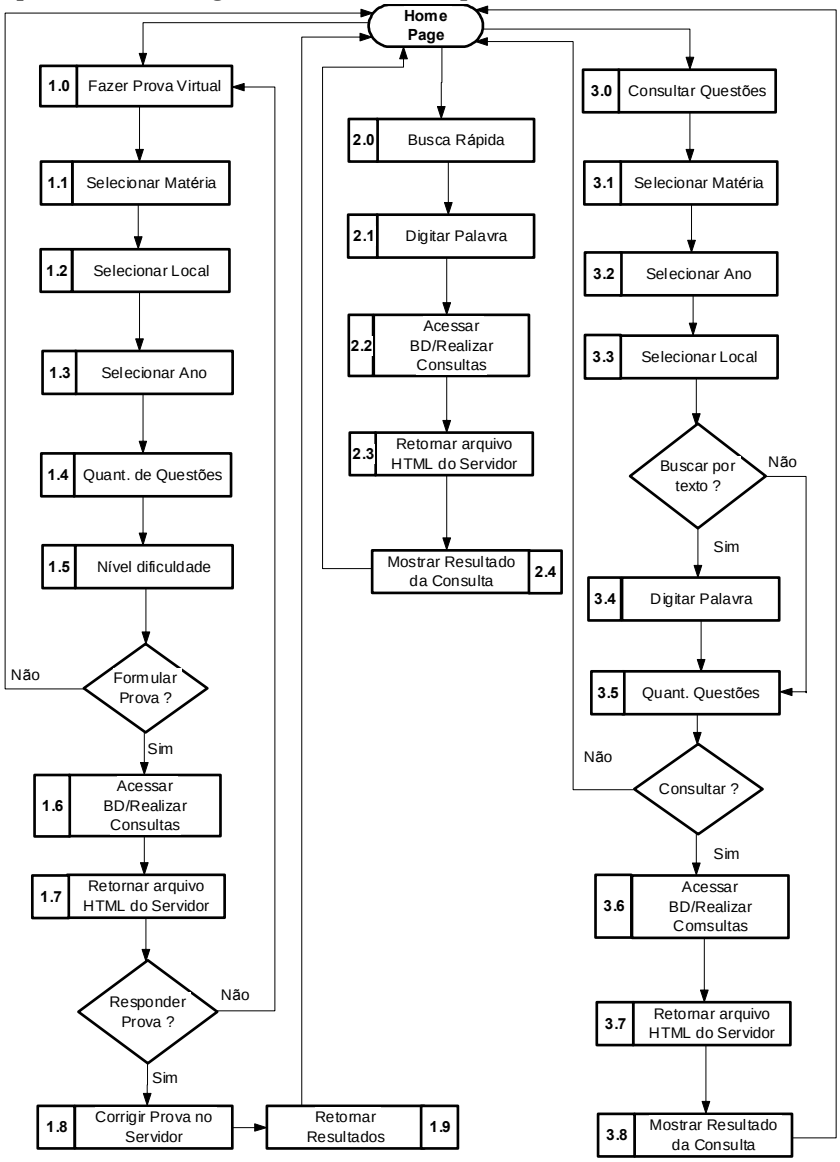

Figura 5.1 - Fluxograma da seqüência de ações executadas durante o processo de visita de um usuário à Home Page

5.1.1. Descrição detalhada dos módulos

A seguir tem-se uma descrição detalhada de cada módulo do fluxograma representado na figura 5.1.

5.1.1.1. Módulo 1 – Fazer Prova Virtual

Aqui está representado o principal ponto da implementação, é onde se dá a partida para que o usuário realize a prova. Tem os seguintes passos:

a) **1.1 – Selecionar Matéria:** uma matéria entre as cadastradas no banco de dados deve ser selecionada.

b) **1.2 – Selecionar Local:** seleciona-se um local de onde a prova é originada, ou seja, onde ela foi aplicada em determinada ocasião.

c) **1.3 – Selecionar Ano:** é onde escolhe-se o ano na qual as questões que formularão a prova foram aplicadas.

d) **1.4 – Quant. de Questões:** representa a quantidade de questões que se deseja na prova.

e) **1.5 – Nível de dificuldade:** é o nível de dificuldade da prova que será formulada, ele pode ser: fácil, intermediário, difícil ou aleatório. Este último formula a prova com um misto dos outros três níveis.

f) **1.6 – Acessar BD/Realizar Consultas:** após todos os parâmetros acima terem sido selecionados, o usuário requisita a formulação da prova e o arquivo *prova.php3* se encarregará de acessar o banco de dados e de posse dos parâmetros realizar as consultas SQL de modo a satisfazer as necessidades do usuário.

g) **1.7 – Retornar arquivo HTML do Servidor:** realizando as consultas todos os resultados da mesma são retornados do servidor na forma de um arquivo HTML, que será exibido ao usuário.

h) **1.8 – Corrigir Prova no Servidor:** com a prova à sua disposição e após responde-la, o próximo passo é tratar da correção da mesma. Isso é feito através do arquivo *corrige.php3*, que enviará ao servidor de banco de dados as respostas dadas pelo usuário e comparará as suas respostas com as corretas.

i) **1.9 – Retornar Resultados:** feita a correção no servidor, os resultados retornarão até o usuário em forma de um arquivo HTML.

5.1.1.2. Módulo 2 – Busca Rápida

A busca rápida dá a opção de pesquisa no banco de dados por palavras chave, ou seja, basta digitar uma palavra para se buscar questões no banco que contenham de alguma forma esta palavra.

a) **2.1 – Digitar Palavra:** é o campo onde a palavra que se deseja será digitada.

b) **2.2 – Acessar BD/Realizar Consulta:** após digitar a palavra e fazer a requisição da consulta, o arquivo *buscarap.php3* acessa o banco de dados e realiza consulta SQL em busca das questões que satisfaçam a condição.

c) **2.3 – Retornar arquivo HTML do Servidor:** ao realizar-se as consultas no servidor um arquivo HTML com os resultados é retornado do servidor.

d) **2.4 – Mostrar Resultado da Consulta:** o arquivo HTML retornado do servidor é exibido ao usuário com o resultado da consulta por ele solicitada.

5.1.1.3. Módulo 3 – Consultar Questões

Neste módulo está representada a possibilidade de se fazer uma consulta mais aprofundada às questões que estão cadastradas no banco de dados.

a) **3.1 – Selecionar Matéria:** seleciona-se a matéria que se deseja passar como parâmetro para realizar a consulta.

b) **3.2 – Selecionar Ano:** consultar questões referentes à determinado ano.

c) **3.3 – Selecionar Local:** consultar questões referentes à determinado local onde a questão foi aplicada.

d) **3.4 – Digitar Palavra:** caso deseja-se consultar as questões também por uma palavra chave, basta digita-la no campo disponível.

e) **3.5 – Quant. Questões:** aqui deve-se selecionar a quantidade de questões que se deseja ser exibida por página no resultado da consulta.

f) **3.6 – Acessar BD/Realizar Consultas:** com todos os parâmetros selecionados, o arquivo *consulta.php3* acessará o banco de dados e processará a consulta conforme os dados passados pelo usuário.

g) **3.7 – Retornar arquivo HTML do Servidor:** após realizar as consultas um arquivo HTML é enviado do servidor contendo as informações resultantes da consulta.

h) **3.8 – Mostrar Resultados da Consulta:** o arquivo HTML que retorna do servidor é mostrado ao usuário com o resultado da consulta.

5.1.2. Descrição do Fluxograma

Inicialmente, tem-se as opções de realizar a prova virtual, consultar questões por palavras e consultar questões por outros campos.

Para a realização da prova virtual, o processo se seguirá com a escolha dos campos a serem requisitados no Banco de Dados para a formulação da prova, são eles: matéria, local, ano, quantidade de questões da prova e nível de dificuldade. Feita a escolha dos campos, pode-se ainda recusar a formulação da prova retornando para a página inicial do Web Site. Por outro lado se houver prosseguimento ao processo realizando a formulação da prova, todos os parâmetros selecionados são enviados ao Banco de Dados em forma de consulta SQL por um arquivo .php3. Este arquivo, por sua vez, realiza no Banco de Dados em questão, MySQL, todos o processos de consulta e seleção requisitados anteriormente retornando um arquivo em formato HTML que será exibido na tela com o resultado da sua consulta. Tendo-se a prova formulada à disposição, pode-se ainda reformular uma outra prova não respondendo-a e retornando à página onde se tem todos os campos à serem preenchidos. Se a prova for respondida, após isso ocorrerá o processo de correção da mesma no servidor e finalmente retornando um arquivo HTML com os resultados.

Na opção de consultar questões por palavras, deve-se simplesmente digitar no campo a palavra que se deseja consultar nas questões armazenadas no Banco de Dados. Com isso o arquivo .php3 irá buscar no Banco de Dados todas as questões que contém a palavra especificada, retornando um arquivo HTML do servidor com o resultado encontrado e exibindo na tela as questões selecionadas através deste critério.

Optando-se por fazer uma consulta um pouco mais criteriosa, deve-se realizar a busca por outros campos além da palavra. A maneira como isso funciona é a seguinte: após escolher esta opção, deve-se selecionar os campos matéria, ano e local de acordo com as necessidades. Em seguida,

existe a alternativa de, além de se buscar por esses critérios já selecionados, consultar também por uma palavra que deseja-se estar presente entre as questões cadastradas no Banco de Dados. Se for decidido fazer a busca desta forma, simplesmente deve-se digitar a palavra no campo e prosseguir com a consulta. Se, por outro lado, a busca por palavras não for utilizada, o processo de consulta segue normalmente com a seleção da quantidade de questões que se deseja por página na exibição da consulta. Com os campos necessários para a realização da consulta devidamente preenchidos ou selecionados, o processo pode ser interrompido recusando-se a consulta e retornando à página inicial do Web Site. Entretanto, se for dado seguimento, a consulta é estruturada em forma de SQL e enviada ao Banco de Dados para ser processada e retornar do servidor um arquivo em HTML que será exibido na tela com o resultado final da consulta.

5.2. Modelagem dos Dados

Agora será relatado como realizou-se todo o processo da modelagem dos dados envolvidos na implementação desenvolvida neste trabalho. Começando com uma descrição das tabelas, passando após para um Modelo E-R, seguindo com um dicionário dos dados e finalizando com a sintaxe SQL utilizada para a criação das tabelas no Banco de Dados utilizado, o MySQL.

5.2.1. Descrição das Tabelas

No banco de dados *vestiba* da implementação tem-se as seguintes tabelas:

a) Questões
b) Resposta

A tabela questões contém as informações sobre a questão que será aplicada na prova virtual, ou simplesmente exibida como resultado de uma consulta. Seus campos são:

Campo	Descrição
numquest	Indica o número da questão no banco de dados. (código)
questao	Contém a descrição da questão que está armazenada, qual é a pergunta.
imagem	Campo reservado para eventual necessidade da questão conter alguma imagem. Só é utilizado se necessário.
materia	Armazena à qual matéria a questão se refere.
ano	Indica em que ano a questão foi aplicada no vestibular.

local	É o local onde a questão foi aplicada.
certas	Determina a quantidade de usuários que acertaram a questão.
erradas	Determina a quantidade de usuários que erraram a questão.

Tabela 5.1 - Descrição dos campos da tabela Questões.

A tabela resposta contém os dados das respostas das questões. Seus campos são:

Campo	Descrição
numresp	É o número da reposta da questão.
numquest	Se refere à qual questão da tabela questões a resposta corresponde.
descricao	Contém a descrição da resposta.
imagem	É um campo utilizado quando a resposta contém alguma imagem, só quando necessário.
certa	Indica se a resposta descrita é a verdadeira ou não.

Tabela 5.2 - Descrição dos campos da tabela resposta.

5.2.2. Modelo E-R (Entidade - Relacionamento)

As tabelas envolvidas no trabalho tem uma relação entre si, que é representada por um modelo e-r. Enquanto a tabela questões contém apenas as suas informações, a tabela resposta, por sua vez, necessita da informação de qual questão determinada resposta faz parte, essa informação é buscada na tabela questões através do campo *numquest* que corresponde ao número da questão.

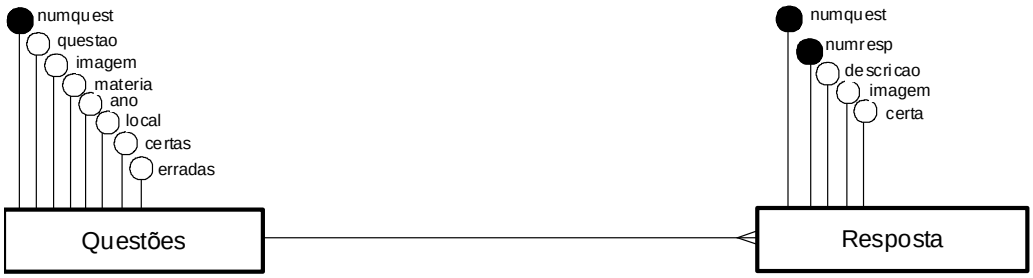

Figura 5.2 - Modelo E-R.

5.2.3. Dicionário de Dados

As tabelas 5.3 e 5.4 representam o dicionário de dados das tabelas questões e resposta respectivamente. Serão apresentados juntamente com todos seus campos as propriedades de cada um, como tipo, tamanho, etc.

Campo	Tipo	Tamanho	Not Null	Extra	Ch. Primária
numquest	Mediumint		X	Auto_increment	*
questao	Text		X		
imagem	Varchar	80			
materia	Varchar	25	X		
ano	Varchar	4	X		
local	Varchar	20			
certas	Float	10,2			
erradas	Float	10,2			

Tabela 5.3 - Dicionário de Dados da tabela questões.

Campo	Tipo	Tamanho	Not Null	Ch. Primária
numquest	Int	11	X	*
numresp	Char	1	X	*
descricao	Blob		X	
imagem	Varchar	80		
certa	Char	1	X	

Tabela 5.4 - Dicionário de Dados da tabela resposta.

5.3. Sintaxe SQL para criação das tabelas

Todas as tabelas da implementação foram criadas utilizando-se comandos SQL. A seguir será mostrado a sintaxe desses comandos durante a criação das tabelas questões e resposta.

a) Tabela questões

```
CREATE TABLE questoes (
```

```
        numquest mediumint NOT NULL auto_increment,
        questao text NOT NULL,
        imagem varchar(80),
        materia varchar(25) NOT NULL,
        ano varchar(4) NOT NULL,
        local varchar(30),
        certas float(10,2),
        erradas float(10,2)
        PRIMARY KEY (numquest));
```

b) Tabela resposta

```
CREATE TABLE resposta (
        numquest int(11) NOT NULL,
        numresp char(1) NOT NULL,
        descricao blob NOT NULL,
        imagem varchar (80),
        certa char(1) NOT NULL,
        PRIMARY KEY(numquest,numresp),
        FOREIGN KEY(numquest));
```

5.4. Descrição das funções PHP utilizadas

No decorrer deste trabalho, houveram várias situações onde funções nativas do PHP tiveram que ser utilizadas. Dentre as quais pode-se citar a necessidade de se gerar números aleatórios para formular a prova aleatória, e para o tratamento das strings na implementação da busca rápida e na busca por texto.

Entre as funções que foram utilizadas deve-se dar destaque também às que se relacionam a banco de dados, mais especificamente ao MySQL, que permitem que se conecte à ele, realize consultas SQL no servidor, entre outras finalidades. Elas serão aqui apresentadas e comentadas uma a uma. As outras funções também receberão um comentário individual, apresentando inclusive a sua sintaxe.

a) **Função:** trim

   Sintaxe: `trim(string str);`

Comentário: apaga os espaços em branco do início e fim de uma string.

b) **Função:** explode

Sintaxe: `explode(string separador, string str);`

Comentário: retorna um array contendo as partes da string com valores separaos por um separador.

c) **Função:** count

Sintaxe: `count(mixed matriz);`

Comentário: retorna o número de elementos de um array. Retorna 1 se a variável não for um array e 0 se a variável não estiver definida.

d) **Função:** strlen

Sintaxe: `strlen(string str);`

Comentário: retorna o comprimento de uma string.

e) **Função:** ceil

Sintaxe: `ceil(float número);`

Comentário: retorna o próximo número inteiro maior ou igual ao número especificado.

f) **Função:** range

Sintaxe: `range(int lim_inferior, int lim_superior);`

Comentário: retorna um array contendo um seqüência de números inteiros no intervalo especificado.

g) **Função:** srand

Sintaxe: `srand(int semente);`

Comentário: altera a semente do gerador de números aleatórios para a função rand();

h) **Função:** rand

Sintaxe: `rand([int limite_inf], [lim_sup]);`

Comentário: retorna um número aleatório dentro do intervalo especificado.

i) **Função:** shuffle

Sintaxe: `shuffle(array matriz);`

Comentário: embaralha os elementos de um array.

j) **Função:** mysql_connect

Sintaxe: `mysql_connect(string[hostname],[usuário], [senha]);`

Comentário: abre uma conexão a um servidor MySQL, retorna *id_link* se tiver sucesso, caso contrário retorna *false*.

k) **Função:** mysql_select_db

Sintaxe: `mysql_select_db(string nome_banco_dados);`

Comentário: depois de estabelecer conexão ao servidor de banco de dados, esta função faz conexão com o banco de dados requerido na função.

l) **Função:** mysql_query

Sintaxe: `mysql_query(string consulta);`

Comentário: trabalha enviando consultas em SQL ao servidor MySQL. Retorna *true* se tiver sucesso, caso contrário retorna *false*.

m) **Função:** mysql_fetch_array

Sintaxe: `mysql_fetch_array(int resultado);`

Comentário: retorna um array que indica o próximo registro do resultado da consulta ou *false* se não houver mais registros no array.

n) **Função:** mysql_data_seek

Sintaxe: `mysql_data_seek(int id_resultado, int num_linha);`

Comentário: durante a verificação dos registros resultantes da consulta SQL esta função move o ponteiro interno dos resultados. Retorna *true* se tiver sucesso, caso contrário retorna *false*.

o) **Função:** mysql_num_rows

Sintaxe: `mysql_num_rows(int resultado);`

Comentário: retorna o número de registros contidos em um resultado de consulta SQL enviada ao servidor.

5.5. Implementação PHP e HTML

O presente item tem por objetivo apresentar a implementação de todos os arquivos .php3 inclusos no estudo de caso, bem como uma breve descrição dos arquivos HTML. Os arquivos serão explicados um a um em suas principais características de implementação.

5.5.1. Index.html – Home Page

O objetivo desse arquivo é ser ponto de partida para as páginas que permitem fazer prova virtual e consulta a questão do banco de dados. Também é possível efetuar uma consulta rápida, através de palavras chaves digitadas pelo usuário.

A chamada ao arquivo que executa a consulta rápida é assim codificada no arquivo index.html:

```
<form action=buscarap.php3 method=get>
<input type=text name=buscar>
<input type=submit value=busca>
</form>
```

Ao ser solicitada a consulta pelo usuário, é requerida a execução do arquivo *buscarap.php3* no servidor, sendo enviada as palavras do formulário como parâmetro para o arquivo PHP. Após execução é retornada a página HTML com as questões que satisfazem as palavras chaves.

Os outros dois links existentes na página servem respectivamente para solicitar a página de prova virtual e consulta de questões. Sua codificação fica assim:

```
<a href="consulta.html">Consulta</a>
<a href="prova.html">Prova virtual</a>
```

Sendo assim essas duas páginas não utilizam nenhuma implementação PHP, sendo arquivos de HTML puro.

5.5.2. Prova.html

Este arquivo é o ponto principal da Home Page, pois será através dele que todos os parâmetros para execução da prova virtual serão enviados ao arquivo *prova.php3* que criará a prova virtual.

Nesta página o usuário poderá optar pelos seguintes parâmetros para formulação da prova:

a) **Matéria** – através de um menu cascata será possível escolher entre as matérias disponíveis, somente uma será selecionada, sendo sua escolha obrigatória.

b) **Local -** através de um menu cascata o usuário poderá selecionar, um dos locais das quais se originam as questões, ou seja qual universidade a questão foi aplicada. Sua escolha é opcional, caso não seja escolhido nenhum local , a criação da prova se baseará em todas as provas dos locais cadastrados **.**

c) **Ano** – através de um menu cascata pode-se selecionar o ano que as questões foram aplicadas. Sua seleção também é opcional.

d) **Quantidade de questões** – nesse menu cascata escolhe-se o número de questões da prova, ou seja dentro das questões encontradas com os parâmetros selecionados, será apresentada para o usuário uma prova com cinco, dez, quinze , ou vinte questões. A seleção da quantidade de questões é o obrigatória sendo a padrão cinco.

e) **Nível Dificuldade** - nesse menu cascata deve-se escolher qual o nível de dificuldade da prova a ser criada. As opções são: fácil, intermediário, difícil e aleatório. O processo de formulação da prova baseia-se nos critério acima mencionados, assim quando o total de questões encontradas com os parâmetros fornecidos pelo usuário é retornada, o sistema avalia dentro de todas as questões, quais as se encaixam no nível de dificuldade solicitado. Esse critério de dificuldade é baseado em dois campos do banco de dados, que contem a informação da quantidade de vezes em que a questão foi acertada ou errada. Essas informações de acerto e erro são alimentadas para o banco a cada vez em que uma prova virtual é corrigida, assim as questões aumentam o numero de erro ou acerto em uma unidade. Para elucidar melhor o processo de criação da prova, segue o exemplo: Suponha que o usuário tenha escolhido o nível fácil. Quando forem encontradas as questões com os parâmetros enviados, o sistema avaliara a porcentagem de acertos de cada questão, assim dentro das questões serão escolhidas as N questões com maior percentual de acertos, sendo essas enviadas para o usuário.

A chamada ao arquivo .php3 que executa o processo de criação da prova, é assim codificado no arquivo prova.html:

```
<form action=prova.php3 method=get>
... menus de seleção dos parâmetros...
</form>
```

5.5.3. Consulta.html

Este arquivo é o responsável pelas consultas que são realizadas ao banco de dados, é nele que se passa todos os parâmetros necessários para a realização da mesma, ele os envia para o arquivo *consulta.php3* que se encarrega de processar as consultas em forma de SQL.

Os parâmetros requisitados neste arquivo são:

a) **Matéria** – um menu cascata existe à disposição do usuário para que se escolha entre as matérias disponíveis qual é a desejada, a opção por uma matéria é opcional. Caso nenhuma matéria seja selecionada, o resultado da consulta abordará todas as matérias cadastradas no banco de dados.

b) **Ano** – a escolha do ano à que se refere a questão também é opcional e é feita através de um menu cascata. Se nenhum ano for selecionado, todos os cadastrados estarão incluídos na pesquisa.
c) **Local** – refere-se ao local onde a questão foi aplicada, também é opcional e em forma de menu cascata. Se nenhum local for selecionado, a consulta se baseará em todos os locais cadastrados.
d) **Busca por texto –** aqui o usuário pode, além de todos os parâmetros mencionados acima, optar por fazer uma busca por uma palavra-chave. Para isto basta digita-la no campo disponível, com isso a consulta será processada considerando a palavra-chave, e somente serão retornadas questões que de alguma maneira contenham a palavra digitada.
e) **Número de Questões** – em um menu cascata deve-se selecionar a quantidade de questões que se deseja por página para o resultado da consulta. Pode-se optar por cinco ou dez questões por página.

Com todos estes parâmetros devidamente selecionados, de acordo com a necessidade do usuário, pode-se dar prosseguimento à consulta com a chamada do arquivo *consulta.php3* que está assim codificado:

```
<form action=consulta.php3 method=get>
... menus de seleção dos parâmetros...
</form>
```

5.5.4. Prova.php3

O objetivo deste arquivo é gerar uma página html que contenha uma prova virtual com questões que atendam aos parâmetros escolhidos pelo usuário. Esse script receberá cinco variáveis oriundas da página *prova.html*, essas variáveis são geradas automaticamente quando o script é chamado, assim ficam disponíveis para utilização na formulação da prova virtual.
As variáveis são:

a) matéria
b) local de origem da questão
c) ano que a questão foi aplicada
d) número de questões da prova
e) nível de dificuldade

Das variáveis acima, a matéria, o número de questões e o nível de dificuldade são de seleção obrigatória, sendo o ano e o local opcionais. Caso o ano e o local não sejam escolhidos, a consulta se baseará em todos os anos e locais cadastrados.

Com toda as variáveis passadas, o script segue-se o processo de geração do código SQL para consulta das questões. A cada variável existe um processo que determina o trecho SQL que deve ser criado, assim ao final de todos os processos tem-se uma string completa com todo o código para ser executado no banco de dados. Somente o número de questões não influenciará na geração do código SQL, servindo apenas para controle das questões que serão exibidas.

Tratamento da variável matéria:

```
$par1 ="materia ='$materia'";
```

A variável *$par1* recebe por exemplo a string *materia = historia*, caso seja esta a seleção feita pelo usuário na página *prova.html*.

Tratamento da variável local:

```
if ($local <> 'np')
     {
       $par2 ="AND local = '$local'";
     }
```

Caso o parâmetro local tenha sido selecionado pelo usuário, ou seja, diferente de *np*, a variável *$par2* receberá a string *and local = URI*, por exemplo.

Tratamento da variável ano:

```
if ($ano <> 'np')
     {
       $par3="AND ano = '$ano'";
     }
```

Caso o parâmetro ano tenha sido selecionado pelo usuário, ou seja, diferente de *np*, a variável *$par3* receberá a string *and ano = 2000*, por exemplo.

Tratamento da variável nível de dificuldade:

Fácil (1)

```
if (($nivel == 1) || ($nivel == 2))
       {
          $par4=" order by porcen desc";
          $cont=0;
```

```
        }
```

Das questões selecionadas, será agregada ao código SQL a cláusula *order*, que ordenará de forma decrescente as questões pelo campo *porcen*. Campo este que contém o cálculo dos campos *certas* e *erradas*, resultando na porcentagem de pessoas que acertaram a questão.

Intermediário(2)

```
if (($nivel == 2) and ($numreg>$numques))
        {
            $cont=Ceil(($numreg - $numques)/2);
        }
```

Como visto no trecho anterior, caso o nível seja o 2 também será incluida a cláusula *order* descrita acima. Porém, dentre as questões selecionadas, e se estas superarem o número de questões escolhidas pelo usuário, a variável $*cont* receberá a posição onde começa a leitura sequencial dos registros.

Difícil(3)

```
if ($nivel == 3)
        {
          $par4=" order by porcen asc";
        }
```

Das questões selecionadas, será agregada ao código SQL a cláusula *order*, que ordenará de forma crescente as questões pelo campo *porcen*. Assim, serão selecionados por primeiro as questões mais difíceis, ou seja, com menor porcentagem de acerto.

Aleatório(4)

```
if ($nivel == 4)
{
$numbers = range(1,$numreg-1); //gera uma sequencia de //numeros
inteiros do intervalo especificado
srand(time()); // incia semente aleatoria atraves do tempo
shuffle($numbers); //embaralha os  numeros contidos no array
$cont = $numbers[0]; //primeira posicao do vetor aleatorio
}
```

// trecho do código que implementa a seleção e impressão das questões em código HTML.

```
if ($nivel == 4)
{
        mysql_data_seek($sqlresult,$numbers[$x]);
```

```
        $x++;
}
```

No primeiro trecho de código, é gerado um vetor com números aleatórios de 1 até a quantidade de registros encontrados utilizando as funções *srand(), shuffle()* e *range()*. Ao final do trecho que gera uma questão HTML, é testado o segundo trecho de código descrito acima, que escolhe a próxima questão à ser impressa através da posição seguinte do vetor aleatório. Por exemplo, suponha-se que tenham sido encontrados 30 registros sendo esses numerados de 1 à 30. A posição 1 do vetor tem o valor 10, então com a função *mysql_data_seek* posiciona-se um ponteiro para o registro 10, e no próximo laço outro registro será selecionado pelo valor aleatório do vetor seguinte..

5.5.4.1. Concatenação das variáveis

Com todas os parâmetros devidamente passados pelo usuário e tratados pelo programa, é formulada a sintaxe da consulta SQL. Isto acontece com a concatenação de todas as variáveis *$parx*.

```
$partodos = $par0.$par1.$par2.$par3.$par4;
```

O valor de cada variável *$parx* é o seguinte:

a) $par0 - Select * from questoes where
b) $par1 – materia = $materia
c) $par2 – and local = $local
d) $par3 – and ano = $ano
e) $par4 – order by porcen asc(desc)

Após estar formado o código SQL, segue-se os passos:

a) conexão com o servidor MySQL

Através da função: `mysql_connect (localhost,usuário,senha)` o script faz a conexão com o servidor MySQl.

b) conexão com a base de dados *vestiba*

Através da função: `mysql_select_db (vestiba)` é feita a seleção do banco de dados a ser usado, nesse caso o banco vestiba

c) execução da query da variável *$partodos*

Através da função: `$sqlresult = mysql_query ("$partodos"),` é executada a instruão SQL contida na variável *$partodos* no banco vestiba, retornando um conjunto de registros, que ficam armazenados em forma de vetor na variável *$sqlresult.*

d) posicionamento do ponteiro para o registro inicial

O objetivo da função: `mysql_data_seek($sqlresult,$cont);` é posicinar o ponteiro para a posição *$cont*(variavel essa definida pela escolha do nível de dificuldade),dentro do conjunto de registros na varivel *$sqlresult*.

e) impressão das questões e respostas

Através de um laço é executada as funções que imprimem o número de questões selecionadas pelo usuário ou as que forem encontradas caso esse número seja inferior ao das questões solicitadas. Seguem-se os passos:

a) Imprime cabeçalho da questão
b) Calcula as porcentagens de erro e acerto da questão através dos campos certas e erradas desse registro.
c) Identifica através de código HTML a questão para futura correção.
d) Imprime os gráficos em forma de tabelas HTML.
e) Executa outra query para seleção das respostas da questão:

```
select numquest, numresp,descricao,certa from resposta where numquest=$numreg
```

f) Gera um formulário em forma de *radio buttons* com as respostas da questão, usando um laço para as número de respostas.
g) Caso o nível de dificuldade seja aleatório, será posicionado o ponteiro para a novo registro a ser impresso, caso contrário é selecionado o registro seguinte.
h) Caso os critérios do laço de impressão de questões na seja validado, é encerrada o laço e finalizado o código HTML.

Assim com esses passos gera-se o código HTML com a prova virtual, sendo agora possível sua correção através dos script de correção de prova.

5.5.5. Consulta.php3

É o arquivo responsável por processar as consultas solicitadas pelo usuário. Após todos os parâmetros terem sido selecionados no arquivo *consulta.html*, o *consulta.php3* trabalhará da seguinte forma:

a) Uma sintaxe SQL com as requisições feitas é montada, para isto todos os parâmetros são concatenados, ou seja, eles são lidos um a um e depois são unidos para formar a sintaxe.

```
  ...
  if ($materia <> 'np')
  {
  $par2 =" materia = '$materia'";
  $par=1;
  }
//nesta primeira condição a matéria selecionada é armazenada em
//$par2.
  if ($ano <> 'np')
  {
  $par4 =" ano = '$ano'";
     if ($par==1)
     {
      $par3=' AND ';
     }
  $par=1;
  }
//na segunda condição o ano selecionado é armazenado em $par4.
  if ($local <> 'np')
  {
  $par6 =" local = '$local'";
     if ($par==1)
     {
     $par5=' AND ';
     }
  $par=1;
  }
//e na terceira o local selecionado é armazenado em $par6.
  ...
```

Para o caso de se realizar uma busca por palavra-chave o processo foi implementado usando as funções trim(), explode() e count(). Após a palavra ser digitada no campo o seguinte código é executado:

```
$buscar = trim($buscar);//elimina brancos no inicio e fim da
//string
$recexplode = explode(' ',$buscar);// cria vetor com todas
//palavras do campo buscar
$contpalavras = count($recexplode);//conta o tamanho do vetor
```

Em seguida, a sintaxe SQL para a busca da palavra desejada nas questões é armazenada em *$parbusca*:

```
$parbusca[$x]="(questao like '% $recexplode[$i] %' or questao
like '$recexplode[$i] %' or questao like '% $recexplode[$i]_')"
;
```

Com isto todas as condições passadas pelo usuário estão consideradas, o próximo passo é unir todas essas condições em uma sintaxe SQL única:

```
$partodos=’Select * from questoes’. $par1. $par2. $par3. $par4.
$par5. $par6. $parbusca[$i];
```

b) próximo passo é tratar da conexão com o banco de dados, primeiramente conecta-se ao banco de dados e após á base de dados especifica.

```
if ($conectou = mysql_connect (localhost,root,php3000))  //testa
//conexão ao banco de dados
if ($conecbd = mysql_select_db (vestiba))  //testa conexão à
//base de dados
```

c) Com a conexão à base de dados estabilizada e com a sintaxe SQL para realização da consulta montada, basta agora executar esta consulta SQL no banco de dados para ter os resultados.

```
$sqlresult = mysql_query ("$partodos");//executa a query
//guardando o resultado em $sqlresult
```

d) Realizada a consulta SQL, já se pode controlar os registros que retornaram do banco de dados mostrando-os na tela.

```
$row = mysql_fetch_array($sqlresult)//a variável $row recebe o
//controle de se ainda há registros para serem mostrados
```

```
print $row ["questao"];//o registro retornado é impresso na
//tela.
```

5.5.6. Corrige.php3

A finalidade desse script é fazer o tratamento do código HTML gerado pelo script prova.php3, assim quando o usuário tiver respondido as questões e solicitar sua correção, será chamado o script corrige.php3 que formará outra página HTML contendo um tabela confrontando as respostas do usuário com as respostas corretas, bem como resultado final de total de acertos e erros. Pelo fato de não haver necessidade de segurança das respostas, foi implementa um solução que mantém dentro da própria página que contem a prova, as suas respostas corretas. Isso é feito incluindo tags HTML que ficam ocultadas para visualização do usuário no browser, assim ao executar o script de correção não é necessário o acesso a banco de dados para consulta e sim apenas o confrontamento da questão respondida com a correta, incluída no próprio código da prova. Porém uma outra rotina é executada, a que permite incrementar o valor dos campos, que definem o nível de dificuldade da prova, assim ao acertar uma questão é somada um valor ao campo certas, e o mesmo ocorre para o caso de erro. Essas informações de acerto e erro, servirão de base para formulação das próximas provas virtuais, conforme nível de dificuldade escolhido.

Esse processo se executa da seguinte maneira:

Ao gerar a prova virtual são criados dois vetores com números seqüências equivalente ao número de questões da prova, são eles *resposta[]* e *correta[]*. No vetor resposta existem um índice que contem a letra da resposta escolhida pelo usuário. No vetor correta[] o índice de mesmo número da resposta contem uma string com a letra correta da questão seguido de sua identificação na banco de dados. Essa facilidade de utilizar vetores é uma combinação da código HTML e da linguagem PHP que permite passagem de vetores como variáveis. Além disso existe outra variável que possui o valor da quantidade de questões da prova, com ela será executado o laço que cruzara a informação dos dois vetores, caso sejam iguais é gerada a rotina do script que imprime o trecho da tabela com a informação do resultado e além disso é executada uma *query* com a função UPDATE que atualiza o valor do campo certa do registro identificado no vetor correta , no caso de erradas ocorre o mesmo processo de impressão porém o campo a ser atualizado é o erradas. A seqüência do código é:

a) Processo de conexão ao servidor e ao banco vestiba para atualização dos dados na rotina de correção.

b) Laço inicial contando o número de questões
c) Extrai-se o valor do vetor correta[], dividindo a string em duas novas variáveis, a com o valor da letra correta ($certa) e outra com o identificador(chave primária) da questão($id)
d) Compara-se a variável $certa com o valor do vetor resposta de mesmo índice.
e) Executa-se o processo de impressão do código HTML com a tabela de resultado
f) Atualiza-se o valor dos campos certas ou erradas através da execução da instrução SQL, baseada no número da questão.
g) Finaliza o código assim que sair do laço.

5.5.7. Buscarap.php3

Aqui é onde ocorrerá todo o processamento da consulta por palavra-chave. A implementação deste arquivo gira em torno de 3 funções: trim(), expode() e count(). Elas foram utilizadas da seguinte maneira:

```
$buscar = trim($buscar);//elimina brancos no inicio e fim da
//string
$recexplode = explode('  ',$buscar);// cria vetor com todas
//palavras do campo buscar
$contpalavras = count($recexplode);//conta o tamanho do vetor
```

Com estas funções as palavras digitadas no campo foram preparadas para a realização da consulta SQL:

```
$sqlresult = mysql_query ("Select * from questoes where questao
like '% $recexplode[$i] %' or questao like '$recexplode[$i] %'
or questao like '% $recexplode[$i]_'″);
```

A conexão deve ser testada em seguida:

```
if ($conectou = mysql_connect (localhost,root,php3000))  //testa
//conexão ao banco de dados
if ($conecbd = mysql_select_db (vestiba))  //testa conexão à
//base de dados
```

Tendo os resultados da consulta em *$sqlresult* o último passo é tratar da listagem das questões encontradas:

```
$row = mysql_fetch_array($sqlresult)//a variável $row recebe o
//controle de se ainda há registros para serem mostrados

print $row ["questao"];//o registro retornado é impresso na
//tela.
```

Conclusões

O objetivo do trabalho foi alcançado, tendo em vista que se buscava verificar a possibilidade de se montar um servidor Web utilizando ferramentas disponíveis na Internet, bem como utilizar as mesmas para a construção de um Site que possibilite acesso à uma base de dados.

Com relação ao banco de dados, inicialmente optou-se pelo PostgreSQL, mas houveram dificuldades de implementação no que diz respeito à conexão entre o servidor de banco de dados e o browser, outro fator que determinou a eliminação do PostgreSQL do trabalho foi a sua escassa documentação. Então passou-se para o MySQL, que mostrou-se uma ferramenta de grande velocidade, porém sem uma grande ênfase no quesito segurança, sendo que seu uso para tal fim não é recomendado pelos maiores especialistas. Além disto, deve-se destacar a facilidade de uso referente à manipulação dos dados, como criação de tabelas, inserção de registros, alteração de dados, etc. Outro ponto importante que fez do MySQL o banco de dados escolhido foi a documentação um pouco mais abrangente, além dos documentos oficiais possui no mercado alguns livros que facilitam o seu aprendizado.

A linguagem PHP facilitou bastante a manipulação com o banco de dados, isto por possuir funções nativas que são específicas para cada banco, no caso do MySQL são mais de 30 funções que dão a possibilidade de se realizar todos os tipos de operações com a base dados através de um browser. Mas, não possui somente funções para bancos de dados, elas abrangem itens como segurança, vetores, strings, etc.

Estas duas ferramentas integradas com o sistema operacional Linux e o servidor Apache Web Server, possibilitaram a implementação do Site descrito no capítulo 5. Esta implementação veio a por em prática todo o conhecimento adquirido no decorrer do presente trabalho, onde pôde-se observar que todas as ferramentas utilizadas tiveram uma integração e um desempenho satisfatórios dentro das características da implementação em questão. Ou seja, o banco de dados MySQL portou-se muito bem por a implementação não necessitar de tratamentos de segurança, mas somente de uma conexão rápida e eficaz, e a linguagem PHP mostrou uma grande familiaridade com o MySQL através das suas funções nativas específicas para este banco.

Sugestões para trabalhos futuros

Como sugestão para trabalhos futuros, fica a implementação de um Site com a utilização do módulo de segurança SSL do Apache, e o espelhamento do banco de dados de uma empresa para a Web.

Referências Bibliográficas

[ARO 00] Aroca, Rafael V., 2000. **"Tutorial MySQL"**, http://www.mysql.com.br, 06/2000.

[ATK 99] Atkinson, Leon. 1999. **"Core PHP Programming"**, Prentice Hall PTR, USA.

[BAK 99] Bakken, Stig S e outros. 1999. **"PHP3 Manual"**, PHP Documentation Group.

[BAR 99] Barreto, Mauricio Vivas de Souza. 1999. **"Tutorial de Linguagem PHP"**, http://www.vivas.com.br, 06/2000.

[CAS 00] Castagnetto, Jesus e outros. 2000. **"Professional PHP Programming"**, Wrox, USA.

[CON 99] Conectiva Informática. 1999. **"Conectiva Guia do Usuário"**, Curitiba.

[FIS 99] Fisher, Herbert G. 1999. **"PHP Guia de Consulta Rápida"**, Novatec, São Paulo.

[JAM 99] Jamil, George L. 1999. **"Linux para principiantes"**, Axcel Books, Rio de Janeiro.

[KAB 98] Kabir, Mohammed J. 1998. **"Apache Server Bible"**, IDG Books, USA.

[RAT 98] Ratsschiller, Tobias. 1998. **"Building Dynamic Websites with PHP"**. http://www.phpwizard.net, 04/2000.

[YAR 99] Yager, Randy Y. e outros. 1999. **"MySQL & mSQL"**, O'Reilly, USA.

"Hypertext Preprocessor", http://www.php.net, 03/2000.

"PHP and MySQL examples and resources", http://www.webrdev.com, 04/2000.

"The resource for PHP developers", http://www.phpbuilder.com, 05/2000.